Calhoun
Institutional Archive of the Naval Postgraduate School



Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations | 1. Thesis and Dissertation Collection, all items

1987-06

A database management system to manipulate data collected at the National Training Center, Ft. Irwin CA

Buck, Stephen D.

http://hdl.handle.net/10945/22720

Downloaded from NPS Archive: Calhoun

DUDLEY KNOX LIBRARY

http://www.nps.edu/library

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

DULLES BOOK LIBRARY
SCHOOL
6002

NAVAL POSTGRADUATE SCHOOL
Monterey, California



THESIS

A DATABASE MANAGEMENT SYSTEM
TO MANIPULATE DATA COLLECTED AT THE
NATIONAL TRAINING CENTER, FT. IRWIN CA.

by

Stephen D. Buck

June 1987

Thesis Advisor: Samuel H. Parry

Approved for public release; distribution is unlimited.

T232268

unclassified
ECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

a REPORT SECURITY CLASSIFICATION: unclassified	1b RESTRICTIVE MARKINGS
a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited.
b DECLASSIFICATION/DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
a NAME OF PERFORMING ORGANIZATION: aval Postgraduate School; 6b OFFICE SYMBOL (If applicable): 55	7a NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
: ADDRESS (City, State, and ZIP Code): onterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000
a NAME OF FUNDING/SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
: ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

TITLE (Include Security Classification): A DATABASE MANAGEMENT SYSTEM TO MANIPULATE DATA COLLECTED AT THE NATIONAL TRAINING CENTER, FT. IRWIN, CA.

PERSONAL AUTHOR(S): Buck, Stephen D.

a TYPE OF REPORT	13b TIME COVERED FROM ____ TO ____	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
laster's Thesis		1987 June	130

SUPPLEMENTARY NOTATION

COSATI CODES: FIELD	GROUP	SUB-GROUP	18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
			National Training Center; database; DBMS; relational model; user's model

ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis provides a step by step development of a database management program. Using a problem defined by the intended user, the thesis develops a model of the real world situation using Entity-Relationship Diagrams. This model is then refined into a Relational Model and implemented into a database management program. The theoretical principles used to validate the refinement process are presented in detail to substantiate the procedure. Included also, are a user specific program which facilitates data manipulation and a user's manual which describes the intricacies of the program.

Ft. Knox's Office of Combat Developments has sponsored this thesis in an attempt to provide a storage medium for data originating at the National Training Center (NTC). The NTC was established in the late 1970's to

O STRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION: unclassified
a NAME OF RESPONSIBLE INDIVIDUAL: Prof. Samuel H. Parry	22b TELEPHONE (Include Area Code): (408) 646-2779; 22c OFFICE SYMBOL: Code 55Py

D FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
unclassified

BLOCK 19 CONTINUATION

meet the need for a more dynamic training facility. The exercises conducted at the NTC have been acknowledged as extremely beneficial to the units involved. However, recent reports by the GAO show that the full potential of the training center is not being realized, as adequate data collection and analysis operations have not been instituted. This has caused units to continually relearn past mistakes and failed to allow units to build upon past performances. In an effort to reverse this trend, several institutions have begun analyzing the data originating at the NTC. This in turn, developed the need for a flexible and responsive data storage system.

Approved for public release; distribution is unlimited.

A Database Management System to manipulate data collected at the National Training Center, Ft.Irwin CA.

by

Stephen D. Buck
Captain, United States Army
B.S., United States Military Academy, 1979

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1987

T 053
38?2
c.1

ABSTRACT

This thesis provides a step by step development of a database management program. Using a problem defined by the intended user, the thesis develops a model of the real world situation using Entity-Relationship Diagrams. This model is then refined into a Relational Model and implemented into a database management program. The theoretical principles used to validate the refinement process are presented in detail to substantiate the procedure. Included also, are a user specific program which facilitates data manipulation and a user's manual which describes the intricacies of the program.

Ft. Knox's Office of Combat Developments has sponsored this thesis in an attempt to provide a storage medium for data originating at the National Training Center (NTC). The NTC was established in the late 1970's to meet the need for a more dynamic training facility. The exercises conducted at the NTC have been acknowledged as extremely beneficial to the units involved. However, recent reports by the GAO show that the full potential of the training center is not being realized, as adequate data collection and analysis operations have not been instituted. This has caused units to continually relearn past mistakes and failed to allow units to build upon past performances. In an effort to reverse this trend, several institutions have begun analyzing the data originating at the NTC. This in turn, developed the need for a flexible and responsive data storage system.

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

I. INTRODUCTION AND BACKGROUND OF THE PROBLEM 9
A. INTRODUCTION 9
B. HISTORY OF FORT IRWIN 10
C. NATIONAL TRAINING CENTER INTO OPERATION 12
D. CONTRACT WITH FT. KNOX FOR A DATABASE PRODUCT 15

II. DATABASE THEORY FOR THE NTC APPLICATION 17
A. CONCEPT USED TO DEVELOP THE PRODUCT 17
B. DATA MODELING 18

III. DESIGN AND IMPLEMENTATION OF THE NTC RELATIONAL DATABASE 29
A. LOGICAL MODEL DESIGN FOR NTC APPLICATION 29
B. PHYSICAL IMPLEMENTATION OF THE PRODUCT 38

IV. RECOMMENDATIONS FOR IMPROVEMENT OF NTC DATABASE 42
A. CHAPTER CONCEPT 42
B. PROGRAM CODE MODIFICATIONS 42
C. SYSTEMS INTEGRATION ISSUES 43
D. NATIONAL TRAINING CENTER RECOMMENDATIONS 44

APPENDIX A: ENTITY-RELATIONSHIP DIAGRAMS 45

APPENDIX B: DATABASE TABLES 51
1. INTRODUCTION AND DEFINITION OF FORMAT 51
2. DATA TABLES 51

APPENDIX C: USER HANDBOOK FOR DBMS PROGRAM 57
1. CONCEPT OF THE USER'S MANUAL 57

2. LOADING THE APPLICATION PROGRAM 58
3. PROGRAM CHOICES AND CAPABILITIES 59
a. Initial Menu of Program 60
b. Secondary Level Menus 62
c. Other Menus used in the Program 65
d. Program Operations 66
4. SUGGESTIONS FOR CONSISTENT PROGRAM PERFORMANCE 76

APPENDIX D: STARTUP PROGRAM 77

APPENDIX E: DATABASE MANAGEMENT PROGRAM CODE 78

APPENDIX F: PROCEDURE FILE CODE 81

APPENDIX G: USER TUTORIAL WITHIN PROGRAM 114

APPENDIX H: OTHER IMBEDDED PROGRAMS USED BY THE DBM PROGRAM 121

APPENDIX I: ADDITIONAL EVALUATION CRITERIA 122

APPENDIX J: CONTRACTUAL AGREEMENT WITH SPONSOR 125

LIST OF REFERENCES 127

INITIAL DISTRIBUTION LIST 128

LIST OF FIGURES

1.1 Fort Irwin, California 11
2.1 Kroenke's Spectrum 19
2.2 Example Relationships 20
2.3 Obligatory / Non-obligatory Relationships 21
2.4 First Transition Move 23
2.5 Normal Forms 25
2.6 Un-normalized example vs. normalized 26
3.1 Unit's Mission Relationship 31
3.2 First Step - relational model for 'Unit's Missions' 33
3.3 Second Step - relational model for 'Unit's Missions' 34
3.4 Final Form - relational models for 'Unit's Missions' 35
3.5 First Step - relational models for 'Unit's Vehicle Use' 36
3.6 Second Step - relational model for 'Unit's Vehicle Use' 37
3.7 Final Step - relational model for 'Unit's Vehicle Use' 38

I. INTRODUCTION AND BACKGROUND OF THE PROBLEM

A. INTRODUCTION

Over the last ten years the United States Army has made a significant investment in both time and resources in an effort to upgrade the fighting capabilities of the country's land forces. The primary vehicle for this training effort was the development of a realistic training environment in the desert of Southern California. This training facility, known as the National Training Center (NTC), was given the mission to upgrade the fighting capabilities of maneuver units through a series of planned and evaluated exercises.

The original intent of the NTC was to provide a set of evaluated scenarios where battalion sized units could be evaluated against an opposing force using soviet tactics. The center would also provide a live fire range where unit gunnery proficiency could be measured. The NTC would present the rotational unit with a total combat environment using a simulated deployment to a Positioning Of Material Configured to Unit Stocks (POMCUS) site where the unit would draw equipment and then conduct simulated combat maneuvers for a two week period. This exercise period would be made up of both force-on-force engagements using the MILES system and live fire missions conducted in a specially designed range complex.

By creating a testing environment away from the home station of the rotational units, the Army hoped for a more consistent evaluation process. Rather than cloud the evaluations with issues dependent upon local facilities, the Army instead developed a training vehicle which sought to make consistent and realistic evaluations of unit proficiencies. As more units were exposed to the center and its unique training environment the Army expected the level of capabilities of its fighting units to progressively increase. High level officers intended to use this method to quickly improve and maintain high levels of combat performance within the country's Active Army component.

B. HISTORY OF FORT IRWIN

Fort Irwin lies on the arid section of California known as Death Valley. The post has been host to several different groups throughout its lifetime and has provided each a unique "sunny" experience. The original Old Spanish Trail was blazed through the area by spaniards on their way to the California coast. Groups of local indian tribes sought refuge there in the early 1800's from the "white man's" expansion into the southwest section of the United States. Kit Carson and John Fremont explored the area and documented the sites which later were used as base camps for units fighting the indians throughout several different periods. The U. S. Army attempted to bring control to the region using an idea adopted from the Middle East when they instituted what was jokingly referred to as the "Camel Corps." Although the idea was sound and the camels adapted to the area, a combination of unfavorable political decisions and the advent of the Civil War brought a halt to the effort. The area also entertained a relatively successful commercial enterprise when 20 mule team trains moved borax from the mines deep in Death Valley out to "civilization." [Ref. 1]

The modern military use of the area began in the 1930's when General George S. Patton used the area to conduct the first large scale armor unit exercises. The post proper was originally opened in August of 1940 as an anti-aircraft gun range and was called the Mojave AntiAircraft Range. It was named Camp Irwin in memory of MG George Irwin, a WWI commander in October of 1942 and proceeded to host armor units training for action in Northern Africa. After the war spread into Europe, the need for a desert training area was lost and the post was deactivated in 1944. As the United States entered the Korean War, the Army once again saw the need for a large scale training area suitable for armor forces. In response to this need, Camp Irwin was opened for military maneuvers in 1951. After the war ended, the explosive potential of large armor units had finally established themselves as a credible fighting force and the need for a suitable training area was accepted. Therefore in 1961, the camp became a permanent post and was renamed Fort Irwin. The post played a significant role in staging units for deployment to Vietnam, but became a victim in the large defense cuts of the early 1970's. In 1972 the site was turned over to the California National Guard and it became primarily a training area for rotational units. [Ref. 2]

The post has provided an excellent staging area for large scale mechanized maneuvers. With its remote location, the post enjoys a respite from interference from civilian or commercial groups. The remote location also allows the combination of

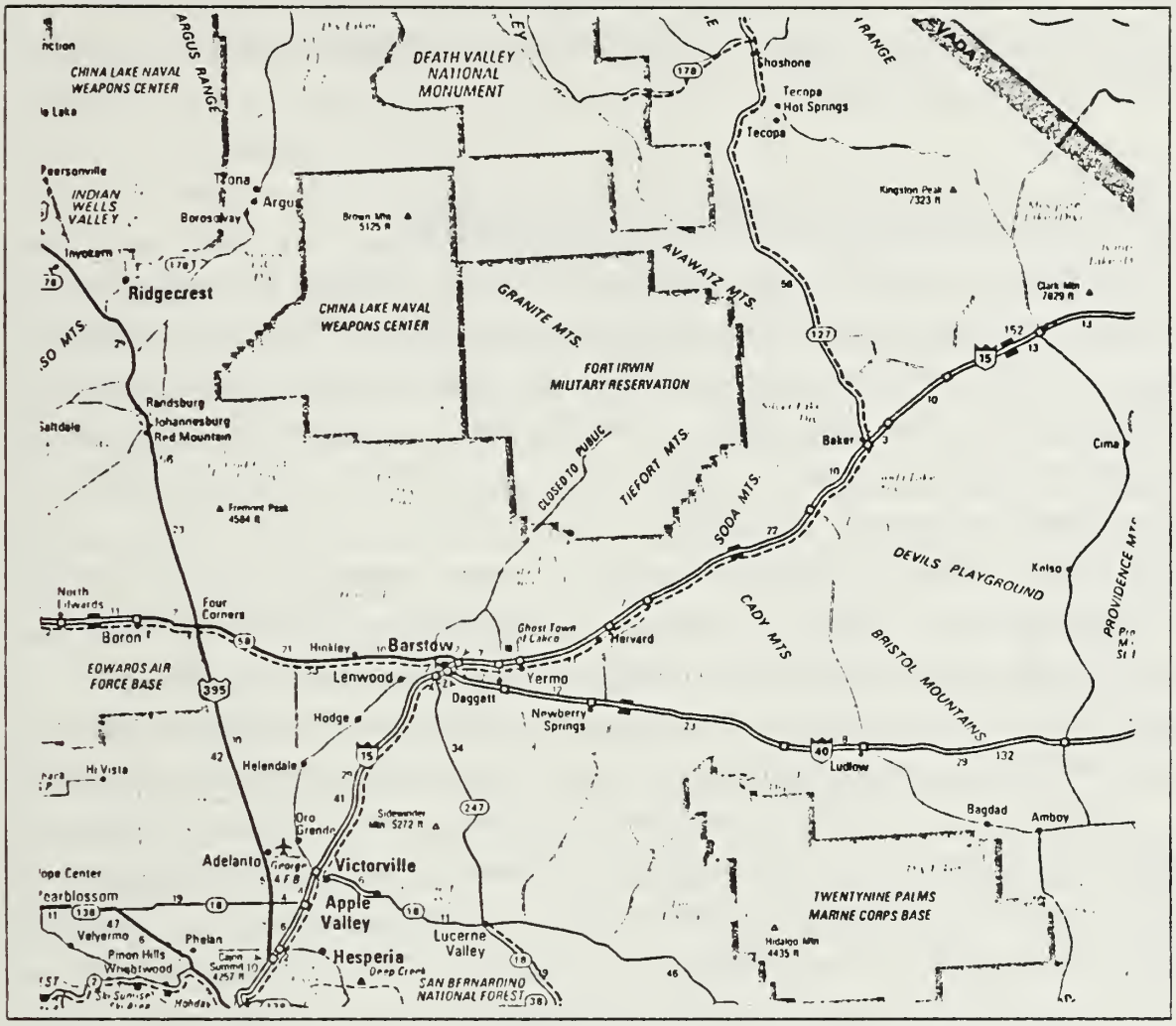


Figure 1.1 Fort Irwin, California.

maneuver and live fire exercises to be held simultaneously. Since the area is so sparsely populated, it provides an excellent facility for armor maneuvers. The post has played host to a variety of Active Army, National Guard, and Army Reserve rotational units which have come to the post to practice armor force techniques. These exercises have run the gamut from live fire exercises to new equipment familiarization.

A special note should be included concerning the post's climate. The arid region most closely represents the Middle East Area. The winters are mild and the summers are extremely hot. The low humidity of the area allows the heat to be bearable, but still a significant aspect for rotational units to plan for. Average rainfall in the area is very small yet the surrounding mountains allow the installation to maintain operation through a vast system of underground wells. Since the Middle East has been host to

several armor conflicts recently, the post has enjoyed a renewed interest. Most experts agree that the installation presents an ideal training environment to prepare for a conflict in that region of the world.

C. NATIONAL TRAINING CENTER INTO OPERATION

The concept for the National Training Center was developed in response to conversations held with senior Army commanders from 1974 to 1976. Many of these commanders felt that the successive wars in the Middle East had shown the need for a renewed interest into armor operations. It was also felt at the time that the world situation favored an outbreak in that region and the United States needed a training area to help simulate actions in that region.

The training facility was first outlined in a concept paper presented in 1976. After receiving favorable reviews, the concept was given high level support within the Army and funding began in fiscal year 1980. Although the project was on a scale larger than previously attempted, the plan did not involve a radical development plan. A similar project on a smaller scale had been successfully demonstrated at Ft. Hood, Texas. Although the Ft. Hood installation used some remote data processing capabilities, the complex at Ft. Irwin required a significant system to be developed and tested for year round use. The Ft. Irwin installation was upgraded to house a large permanent opposing forces group (OPFOR) and a significant facility upgrade was also undertaken. When the facility opened for rotational units in 1981, the post incorporated the most advanced large scale unit training facilities in the free world. [Ref. 2]

The total funding appropriation for years 1980 through 1987 has surpassed one billion dollars. For that cost the Army has received a training facility which can successfully measure a significant portion of the tactical maneuvering of battalion sized units. When this is coupled with the unit evaluation teams, the facility provides a complete and thorough evaluation of CONUS based rotational units. The evaluations are based upon data collected by a wide network of remote sensors which track the maneuver elements of both the OPFOR and the rotational units and unit evaluator input. Recorded measurements of engagements, vehicle locations, and movement speeds allow an after-action critique of each mission. In the live fire area, computer controlled targets simulate an attacking Soviet Motorized Rifle Regiment. The targets,

sensitive to both live rounds and MILES firings, provide the units with realistic engagement scenarios through computer simulations. Upon completion of the exercise period, the unit is provided with a final debrief and a take-home package to use to remedy areas of weak performance.

A normal rotation involves a period of approximately five weeks. After deployment details and pre-rotational briefings have been held, the unit arrives at the NTC and over the period of a week to ten days draws, inspects and prepares its vehicles and personnel for the graded exercises. The force-on-force and live fire exercises last approximately fifteen days and follow two scenarios. The first scenario has one unit conduct force-on-force exercises for approximately five days, then conduct the live fire training for four days, and finish with force-on-force exercises. The second scenario has a unit complete all force-on-force training before moving to the live fire exercises. At the completion of these exercises the unit returns to the post proper, receives its final critique, turns in the drawn vehicles, and returns to its home station.

The evaluated exercises consist of a number of standard missions which a unit might be called upon to perform in a combat situation. They include a defensive position, attack, counterattack, delay, hasty defense, and night operations. In most situations, the units perform one mission a day except for the defensive position. In that mission the units are given a longer period so that they can prepare realistic positions which are dug in and well concealed within the allotted terrain. The live fire missions are a day attack and a day and night defense. The only significant difference between these are the safety requirements imposed by the use of live ammunition.

A unit is evaluated using a list of seven operating systems which has been developed to check all areas of unit operations. Unit evaluators are attached to each unit and subunit down to the level of the platoons. Staff level evaluators maintain their position next to the individuals whom they evaluate while the maneuver unit evaluators trail their units in M551 track vehicles to better assess the action. At the end of each mission, the maneuver units are consolidated where the action stopped and local critiques are conducted. Once platoon critiques have been completed, the company as a whole is evaluated by the senior evaluator assigned to that company. Finally, the commanders and staff of the battalion are critiqued upon their performance and the mission success or failure is assessed. A recording of each critique at the battalion level is made and included in the unit's take-home package.

The battalion level critique is conducted with the aid of a time incremented recording of the mission just completed. The system of remote sensors allows the evaluation group to track the movement, vehicle engagements, and radio traffic of both the OPFOR and the rotational unit, or BLUE force. Transmitters mounted on all vehicles send signals to the remote sensors which are collected, analyzed, and transformed into meaningful output to be used in the critique. The evaluation takes place in a mobile unit which allows the evaluators to "play back" items of interest on a large screen projection. Using the actual movements, the unit is critiqued upon its ability to move, shoot and communicate effectively throughout its mission. The "eye in the sky" misses nothing and provides a most thorough view of what really transpired during each mission.

Once the mission critiques have been completed, the battalion is informed of its next operation and the planning process to develop a mission order begins. Prior to the execution of that order, the unit must reconstitute its forces in a similar process as units would in combat. Requisitions are made to request "new" personnel, vehicles, and supplies to compensate for those "lost" or "used" during the last mission. Should a unit not complete the reconstitution process, it fights with what ever force it has. Many times the unit's success or failure depends upon the administrative actions that preceded the battle. This step along with the in depth evaluation process is what separates the training at the NTC from training conducted at any other installation. The training is designed to be strenuous and tough and only units which are able to accomplish all aspects of combat are considered successful.

The turn-in process allows the unit to repair any vehicles damaged during their rotation to the best of their abilities. As the supply system can not always meet the needs of the user, a civilian contractor maintains overall control of the rotational vehicles to ensure that the vehicles maintain operational standards. During this phase, the unit is debriefed and provided with a historical summary of its performance. In an effort to maintain objectivity, no unit is compared to any other but against a standard scale. Bragging rights are most certainly determined based upon this scale once the unit returns to its home station.

The take-home package presented to the units for use at their home station includes a written summary of the events of their rotation and video tapes of the critiques. There is also limited footage taken of the actual battles where recording was possible. The written report highlights each mission in terms of key events and then

present a summary of the status of the battalion at the completion of the mission. The status report shows vehicle kills, personnel losses, engagement summaries by weapon system and radio communication traffic. The video tapes are keyed to the critiques and attempt to focus on the interaction of the evaluators and the unit in the hope of illuminating weak areas. Although the evaluations are intended to specifically show areas of weakness, they also report areas of strength. This provides the unit with an honest appraisal of its true fighting capabilities.

D. CONTRACT WITH FT. KNOX FOR A DATABASE PRODUCT

Although the National Training Center provides what is claimed to be the best training experience known, recent GAO reports have indicated that the full potential of the NTC is not being realized. In particular, returning units are not performing significantly better than first time participants. The GAO has found that the information collected during the rotations was not being fully disseminated to other rotational units by any agency. The GAO also criticized the Army for not fully disclosing what lessons were learned by each rotational unit. In effect, this caused units to relearn past lessons because units repeatedly made the same mistakes. [Ref. 3]

While in many aspects the GAO report is accurate, it does present a somewhat unrealistic appraisal of the data collected at the NTC. The report concludes that the data collected at the NTC is incomplete for an in depth analysis because it lacks the ability to provide an action by action description of a mission. [Ref. 3] This is true as the present system cannot fully describe a vehicle's location since it collects data using only two axis. This allows it to appear that vehicles can engage other vehicles when they are in fact masked by the terrain. Since terrain elevation is not included within the position coordinates, judgemental decisions are required in the final analysis of vehicle engagements.

This problem has been resolved through the awarding of a contract to modify the system to include the collection of positional elevation. However data collected prior to that point was adequate for trend analysis. The point that eluded the GAO report was that the data collected at the NTC was not designed to evaluate individual actions, but to confirm or disprove trends within the missions as a whole. To attempt to "lock in" on key data attributes and explicitly cite specifics is unrealistic because of the nature of any combat exercise. The bottomline of this analytical exercise was to

introduce or disprove exercise trends for follow-on units to use, but not to dictate the specifics of how a unit accomplished a mission.

Other modifications to the system include the addition of tactical air players to the groups which are controlled by the Multiple Integrated Laser Engagement System (MILES) system and a better implementation of field artillery effects. MILES is the engagement system which allows individual players to "shoot" one another. Remote area sensors monitor these "firings" and record them as tactical input data to be used in the unit evaluation. The addition of the air assets into this engagement system will allow a more in depth appraisal of unit actions since a more realistic combat environment will be presented to the rotational unit.

In an effort to incorporate the lessons learned from experiences at the NTC, several different agencies attempted to analyze data originating from the NTC. The Office of Combat Developments at FT. Knox, KY was one such agency. However within this process of analyzing the data was an inherent need to provide a capability to store relevant data. This thesis is based upon the need to develop a storage mechanism for data from the NTC to be used by an analyst at a distant site.

In an effort to define the nature of the need, a contract of proposed work was developed. The contract, provided as Appendix J, outlined the fact that a need did in fact exist and a solution was possible. It then outlined in very general terms, what system would be developed and tested. The system would also be constructed to utilize a degree of confidentiality to prevent disclosure to persons not authorized to know such details.

Other items of interest within the contract include the need to develop a system which was capable of stand alone operations. This allows the user to float his work between several different locations. The system was requested to focus upon a product which could be utilized in a microcomputer environment. Again, this represented a need for system portability. Finally, the system should require limited maintainance needs as there existed a limited level of expertise at the user's location. Since the thesis due date was in June of 1987, a similar delivery date was given to Ft. Knox.

II. DATABASE THEORY FOR THE NTC APPLICATION

A. CONCEPT USED TO DEVELOP THE PRODUCT

The problem posed by the Office of Combat Developments revolves around the collection of data developed by rotational units at the National Training Center. The user wishes to store the large amounts of data in a permanent mode for analysis at a future date. In a simplistic application, this can be visualized as a large file system which provides the user the ability to access and manipulate large amounts of data in an efficient manner. Such an intention lends itself to a database application.

The fundamental advantages of a database system would easily apply to this application. The creation of the database would allow the data to be organized and stored in a compact means. The system, by definition, allows the user to quickly access the specific information with which he is concerned, and allows him the ability to update the information conveniently. Finally, the database application, if implemented correctly, allows the database to be purged of unnecessary duplications and can avoid inconsistent data entries by reducing the number of times the entry must be made.

Presently, data storage costs have been reduced to a fraction of the cost of the overall computer system. As such, the question of data storage has evolved from one of what data do we store to one of how do we store the data. This allows the designer to concentrate on a logical and concise method of data storage in the quest to solve his problem. This process encompasses two distinct phases: the logical representation used to model the real world application, and the physical design which incorporates the logical representation into physical constructs using a Database Management System (DBMS). The thrust of this chapter will be to describe the design sequence utilized in this thesis.

This chapter will attempt to provide a logical discussion of why the respective options were chosen and how the specific components were designed. The design process was started with Entity-Relationship Diagrams since they adequately modelled the application. These diagram representations were refined by implementing them

into table forms using the relational data model. The tables developed by the relational model were then normalized to remove unwanted duplications. At this point the tables were directly incorporated into a DBMS using a commercially available product. It also appeared that to continue the reduction process would be of little gain, since the final objective was to achieve a form which could be directly implemented into a DBMS.

B. DATA MODELING

To effectively construct any database system, the researcher must first design a model to represent the real world application. Inherent in this modeling process is the model's ability to relate its configuration for data display in such a way as to easily allow further refinement or direct application into a database management system. Kroenke describes a series of data models that allow the designer to transcend the spectrum from a strictly logical perspective to a machine oriented implementation. [Ref. 4: p. 193] This chapter will use his sequential relationships between models within the discussion to highlight the movement from model to model.

To develop a logical representation, the researcher must first model the situation in such a way as to allow it to be intuitively acknowledged by the user. To accomplish this task, the researcher begins the development process by describing the data in easily understood terms (with respect to the user's background and capabilities). Kroenke's spectrum shows a range of focus for each model in terms of its applicability to either a human or machine construct. Using this spectrum, the researcher can start on the left side of the spectrum and sequentially refine the design until he reaches a state where direct application into an available DBMS is possible. This sequence also allows the user to understand the design process since the refinement follows a logical progression of data reduction. This chapter will provide a sequential movement through Kroenke's Spectrum (Fig 2.1), .developing each model within the discussion, until a point is reached where a logical move can be made to an implementable form.

Kroenke's first model is the Semantic Data Model (SDM). This model reflects an entirely verbal description of the data and the relationships between them. Although this model presents a clear description of what the designer intends, no DBMS has ever been designed to strictly implement the verbal descriptions of SDM. [Ref. 4: p. 194] SDM also lacks the concise, formalized data relationships which allow

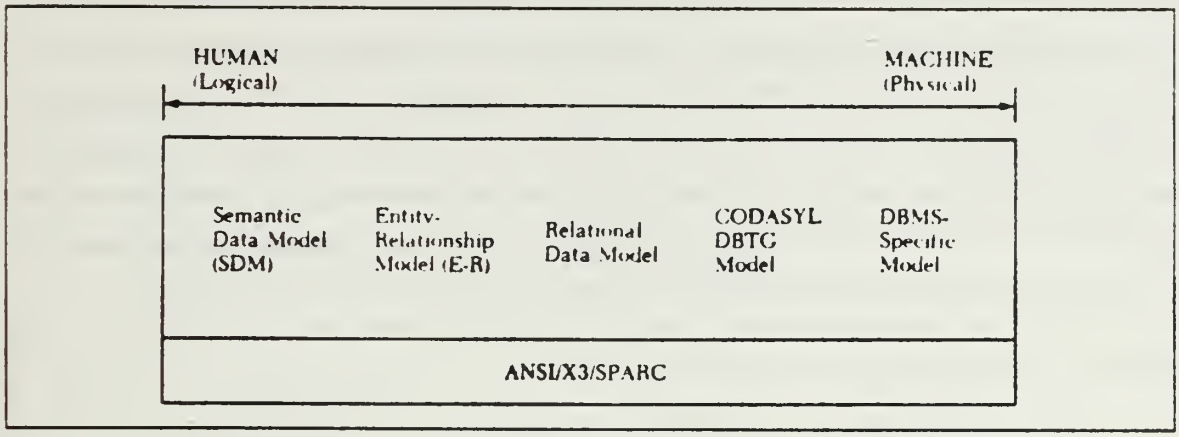


Figure 2.1 Kroenke's Spectrum.

easy refinement into an implementable form. Therefore, for the scope of this problem, the design process was not initiated with it. However, should the user be less familiar with the acronyms and descriptors used to refer to the data items, this model would facilitate that understanding.

The Entity-Relationship Model (ER) provides a more precise interpretation of the individual data items and their relationships. Modell, in a paper presented to the IEEE, referred to the Entity-Relationship Diagram model as "one of the most effective methods of holistic analysis (of the real world)." [Ref. 5: p. 123] Modell concluded that the analyst was able to construct a meaningful model of the real world based upon his interpretation of it. In fact, Modell feels that the only limitation of the diagrams is the user's ability to recognize the different relationships and entities. In light of this, the Entity-Relationship model is the most appropriate tool to begin the design process.

Briefly, the ER model is made up of diagrams which are used to model the real world situation. Each object in the real world can be described through a series of traits or Attributes. An Entity is used to represent this object by consolidating the attributes into a singular body. Groups of similar objects or Entities are collected into Entity Sets.

Entity Sets which are associated in some ways are said to have a relationship between themselves. A relationship can also have attributes which are descriptive of the association of the two entities. There are three distinct types or degrees of relationships possible in ER diagrams. A single entity which is associated to one and only one entity of a different entity set is said to have a one to one relationship (1:1). An single entity which can be associated with more than one entity of another entity

set is said to have a one to many relationship (1:M). Finally, when multiple entites can be associated with multiple entities of a different entity set they are said to have a multiple to multiple relationship (M:N). The significance of these different relationships is that they allow a convenient represention of the interdependence of different objects in the model. A more colloquial understanding of entities and relationships can be described as: entities refer to nouns and relationships refer to verbs. An illustration of the three relational degrees follows (reference Figure 2.2).

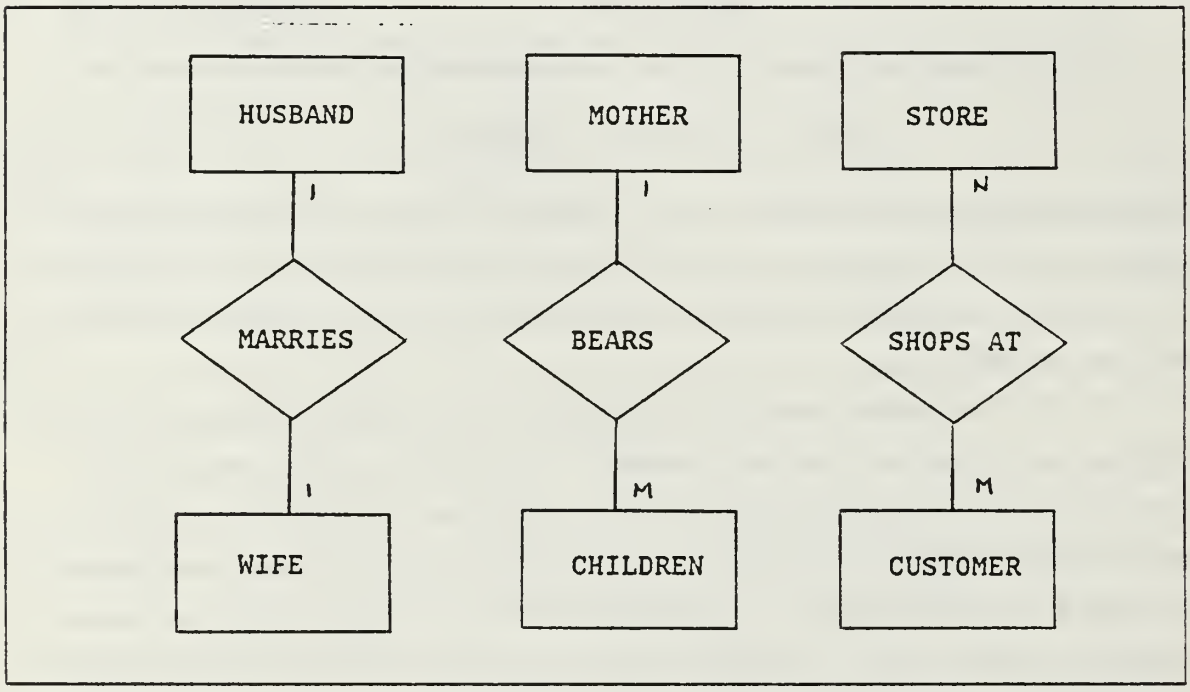


Figure 2.2 Example Relationships.

A further descriptive refinement of the relationships between entites has been defined by the terms "Obligatory" and "Non-obligatory." These terms refer to the membership rules which relate to each member of the entity set. The term "Obligatory" refers to a situation where each member of the entity set must be included in the relationship in order to exist. An example of this would be: before a car can exist it must be manufactured. Therefore in the relationship between manufacturer and car, the car has an obligatory relationship. Conversely, when an entity can exist without inclusion in the relationship, the relation is said to be "Non-obligatory." Again in the manufacture example, a manufacturer can exist without making cars. Therefore, its role is said to be non-obligatory. These new terms are represented in Figure 2.3

below as additional marks on the diagram. The significance of this added description of the relationship is realized during a later time in the development process. The requirement for membership by one or both of the entity sets allows the researcher to consolidate the information from the entity sets and the relationship during the movement process from the ER diagrams to a subsequent model. This in turn allows the move to be a relatively simpler one. [Ref. 6: p. 127]

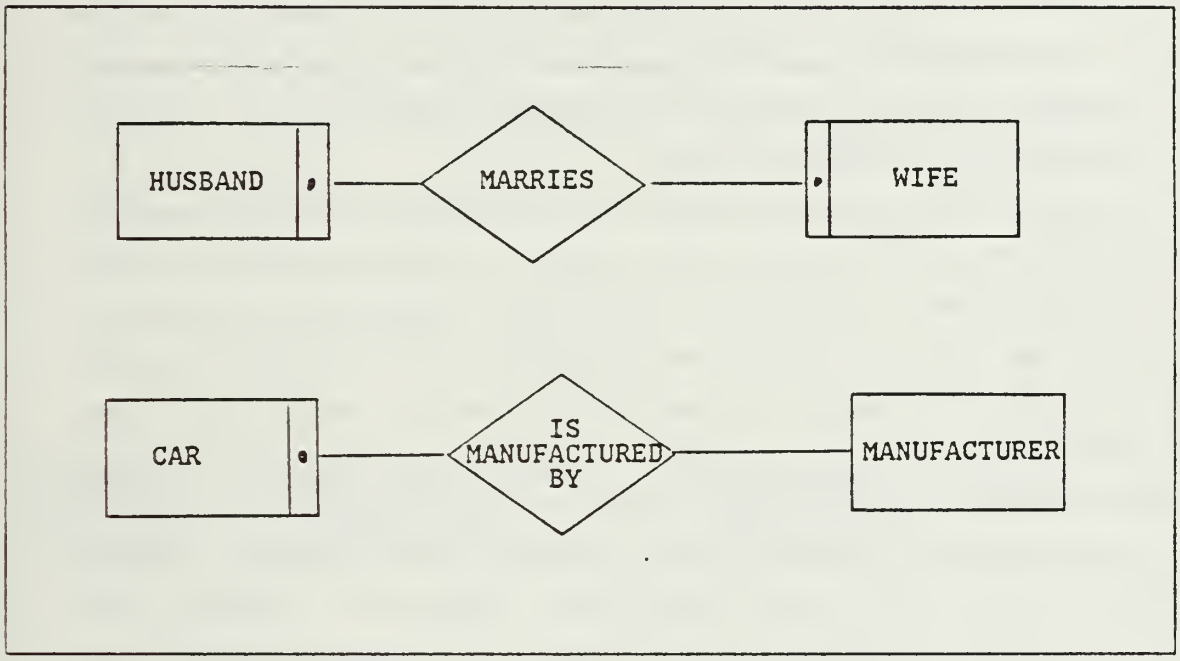


Figure 2.3 Obligatory / Non-obligatory Relationships.

Within each entity there is a specific attribute which can uniquely differentiate this object from others in its respective set. This attribute is commonly refered to as the Entity Identifier. Since a relationship is the association of entities identified by entity identifiers, the relationship can utilize these same constructs. These multiple identifiers within each relationship are refered to as a Relationship Identifier and are nothing more than the consolidation of the two identifiers of the respective entity sets.

Once the data has been completly modelled by the Entity-Relationship Diagrams, the researcher is ready to take the next step in the design process. Using Kroenke's spectrum as a guide for the progression (see Figure 2.1), the next step takes the entity-relationship diagrams and refines them into a relational model. This is a necessary step since as of yet there exists no software program which can take the entity-relationship

diagrams and implement them into physical DBMS constructs. [Ref. 4: p. 228] The relational model will provide a closer link to an easily implementatable form.

The relational model allows the researcher to refine the database model into a more practical form. Since the model was originally evolved from the concept of set theory, the representation maintains a logical approach to data collection. [Ref. 7: p. 19] This allows the user to easily follow the transition from ER diagrams to the relational model. The groupings of the data will tend to relate to sets in the same way as the the Entity Set was a collection of Entities. Operations to manipulate the data can also be modelled after set theory operations. This transition to the relational model represents a move to a location in Kroenke's spectrum that is a midpoint between the logical and physical constructs.

The relational model represents data in a two-dimensional table called a relation. Within the relation, data is stored in rows or tuples. The fields of a tuple are called attributes and represent the columns in the table. These attributes directly relate to the attributes that are used to describe the entities. The number of attributes or columns within a table defines the degree of the table. A table with n columns is said to be a relation of degree n.

Data is normally stored within the table from top to bottom, left to right. No two rows are allowed to be the same within a relation. The first column or columns are generally reserved for the keys of the relation. The key of a relation is taken directly from the Entity Identifier in the ER Diagrams and provides the same function of a data discriminater as in the diagrams.

In moving data from the ER Diagrams to the relations the following steps are initially taken. First, a table is allocated for each entity set and relationship within the diagrams unless an obligatory relationship exists. In those situations the obligatory entity sets are combined in the table corresponding to the relationship. If only one entity set is involved, the result is two tables. If both entity sets are involved, one table can be used to describe the relationship. [Ref. 6: p. 127] Next, using the entity sets, a tuple within a relation is allocated to be filled with the data of each entity within the set. As previously stated the Entity Identifier of the entity set becomes the Key of the relation. For each relationship the relationship identifier becomes the Key (which also should be the consolidation of the two keys of the appropriate tables). Any corresponding attribute from the relationship should be placed in the tuple corresponding to the key.

In the relational model it is important to note that in this first step of moving data from the ER Diagrams to the relations, the relations corresponding to the entity sets will have one or more keys. The relations corresponding to the relationships of the ER Diagrams will have at least two keys. A representation of the move is shown in Figure 2.4.

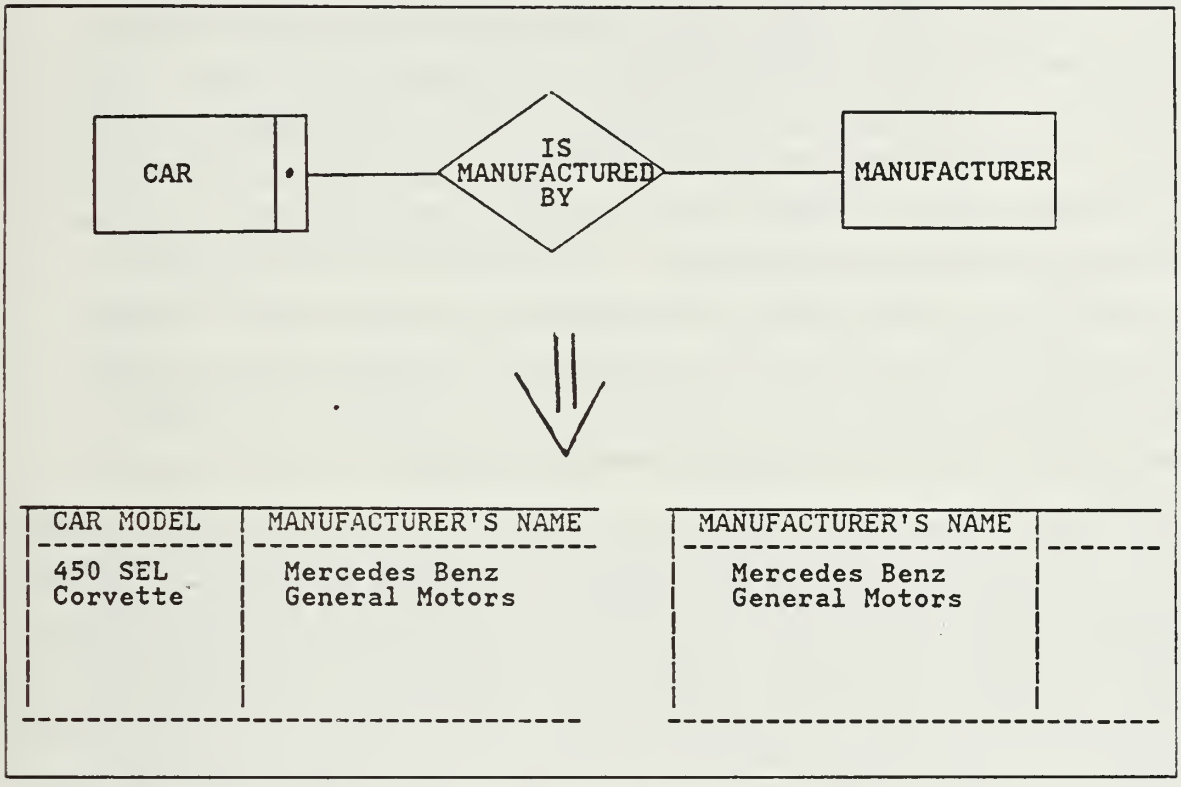


Figure 2.4 First Transition Move.

When the transition process has copied all the data into the relations, the refinement process for this model can begin. This process is refered to as normalizing the tables and removes the unexpected consequences of changing data from the ER diagrams into the relational tables.

The normalization process begins with a basic redistribution of values throughout the relational tables. The first step is to duplicate tuples to eliminate multiple entries in attribute columns. This step places one and only one value within each tuple's attributes. At this point the tables are said to be in their lowest form or at an atomic level, which refers to the fact that the table values cannot be broken down any further. Using this process a null or nonexistent value is allowed to be placed in locations

where a value is unknown or not applicable. This step is normally a trivial task, but it constitutes a starting point for further work.

At this point process of moving the relational model through the series of normal forms is initiated, which will allows the generation of a more compact and redundant free data model. It is of significance to note that presently there is contention over the ability to normalize the ER diagrams. Specifically, the idea is disputed by Chung, et al and Ling in recently published papers. Chung's paper concludes that the normalization process is so restrictive when applied to the diagrams that it stifles the creative abilities of the diagrams. [Ref. 8] Ling proposes a method to normalize ER Diagrams which attempts to counter the complaints identified by Chung. Although his methodology follows a logical schema, it appears that he imposes a limitation upon the initial descriptions of his entities and relationships. In other words, the process appears to initially limit the ways that an entity or relationship can be defined, thereby restricting membership within the sets. [Ref. 9] Since normalization is approached from a broader perspective when implemented within relational tables, the design will follow the recommendation of Chung and postpone normalization until the relational model is constructed.

Over the past decades researchers in relational table theory have devised a series of normal forms which describe the relationships of data within the relational tables. Kroenke gives an illustration which is most useful in understanding the different levels (see Figure 2.5). [Ref. 4: p. 288] The importance of this refinement process is not the actual movement through the normal forms but what each form means. The focus of the process is to develop tabular relations that are the most complete, logical, and redundant-free tables possible.

To better understand this process, several key terms must first be described. A value (A) within a table is said to be functionally dependent upon another value (B) if the value which A can have in some way is dependent upon the value of B. In other words, the values that can be assigned to A are chosen based upon what value has been assigned to B. An example of this is grocery items in a supermarket. The price of the item (A) is assigned (or dependent) based upon the brand of the item (B). In the future we will shorten the description to a statement such as: item (B) --> price (A) or B --> A. [Ref. 4: pp. 289-290]

The next critical piece of information is the idea of a determinant. A determinant is the value which is the cause of the functional dependency. In other words, using the

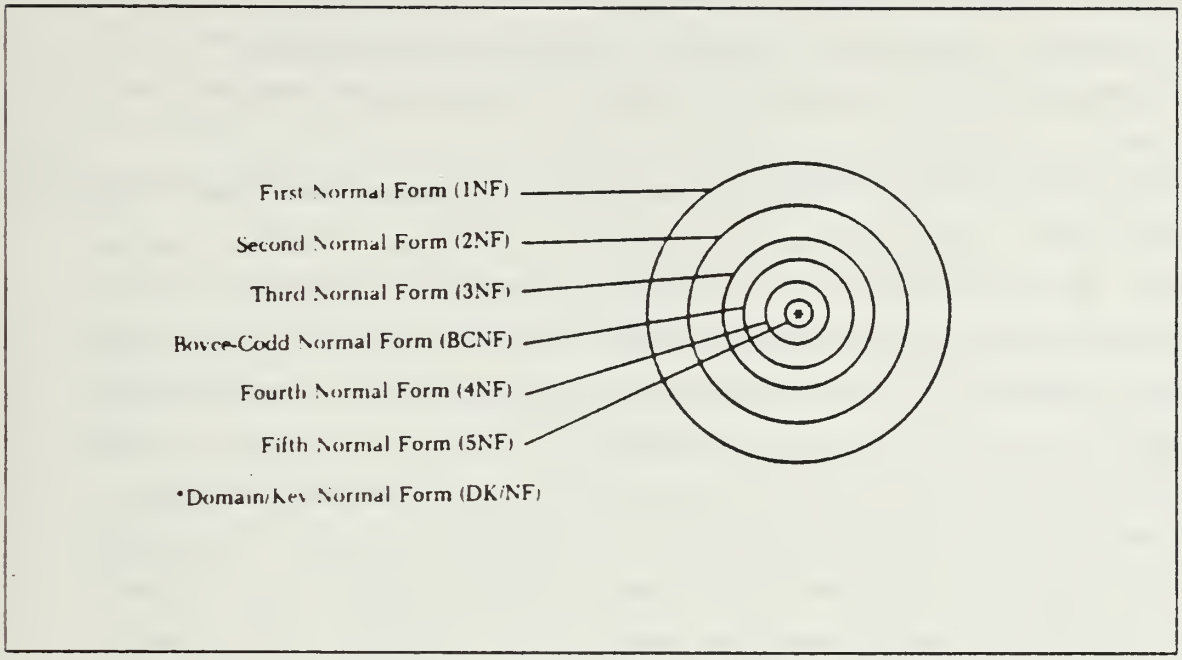


Figure 2.5 Normal Forms.

previous example, the item was the determinant of the price because it determined which value (of price) was assigned. Determinants and functional dependency are very closely linked and can allow a strict interpretation of the relationship between columns in the relational tables. Determinants, however, need not be restricted to single attributes. One value can be the determinant of different attributes and a value can "inherit" a determinant from a value which is its determinant in a transitive relationship. An example of a situation of transitive determinancy is (back to the supermarket): the item (A) determines the price (B) which in turn determines the sales tax (C). In this example A is said to directly determine B, B directly determines C, and A transitively determines C. therefore, C is said to be transitively dependent upon A.

Now that the important terminology has been explained we are ready to define the appropriate normal forms. By eliminating all the tuples which had multiple entries in a column, we have placed the tables in first normal form. This is regarded as the starting point for further work and basically refers to any relational table. Second normal form refers to the fact that any attribute which is not a key is dependent upon all of the keys which define that tuple. Third normal form refers to tables which contain no transitive dependencies. The form with which this thesis will finalize the modelling process is Boyce-Codd Normal Form and refers to the fact that every

determinant in a table is a key for that table. Forms four, five, and DK/NF are more restrictive forms which control functional and join dependencies, yet in an implementation, result in requiring more table additions than they save. [Ref. 4: p. 305]

Once the tables are in BCNF, they have satisfied the attempts to attain a fully normalized model. This process has now provided a model that is simplier to use and work with. It also has eliminated the majority of unwanted duplications within the tables, which in turn, allows the minimal number of entries possible to be used during an update operation. But there are two less obvious advantages to performing this task. Once this task has been accomplished, the user is free to insert or delete information without worry of trying to ensure that other information either exists or will be retained. These are called deletion and insertion side effects. The problem arises when the user is confronted with a situation where if he had deleted a tuple he would have lost information concerning another object. [Ref. 6: p. 85] Using the supermarket example (see Figure 2.6): if he had deleted the tuples in the top table, he would have lost information concerning the manufacturers. By using normalized tables, once an item is sold out, he can delete it from the inventory table yet retain the pertinent information about the manufacturer which would allow him to reorder it.

ITEM	PRICE	SALES TAX	MANUFACTURER	LOCATION	QUANTITY R/O
juice	$1.37	$.06	General Mills	Wisconsin	50
milk	$1.99	$.10	Holly Farms	New York	235

ITEM	PRICE	SALES TAX
juice	$1.37	$.06
milk	$1.99	$.10

MANUFACTURER	LOCATION	QTY	ITEM
General Mills	Wisconsin	50	juice
Holly Farms	New York	235	milk

Figure 2.6 Un-normalized example vs. normalized.

complicated combinations of their predecessors, the most reasonable choice appears to one of the latter categories. Ullman expresses this view by saying that most present languages are complete in that they simulate all the features of relational algebra and relational calculus. He also points out that "In truth, data manipulation languages generally have capabilities beyond those of relational calculus." [Ref. 7: p. 174]

III. DESIGN AND IMPLEMENTATION OF THE NTC RELATIONAL DATABASE

A. LOGICAL MODEL DESIGN FOR NTC APPLICATION

As described within the Data Modeling section, the Entity-Relationship Diagrams are favored as the most practical model from which to initiate a design. The diagrams allow the analyst to model the real world situation confronting him in such a way as to capture the true data associations. The diagrams present representations of the critical players involved in the situation and their relationships to each other. By first developing diagrams to model the situation, the researcher can filter out the immaterial aspects of the problem. This allows him to highlight those areas of interest, but remain flexible enough to add items to the diagrams without compromising them.

In beginning the Entity-Relationship Diagrams, the objects that would be reported by the National Training Center were modelled as entities. This relatively long laundry list of items required additional features to ensure their ability to remain distinct within their respective sets. In many cases that inferred the creation of specific identifiers to accomplish this. Most notable was the creation of a mission code number which was comprised of the year of the rotation, the home station of the rotational unit, and a number which refered to whether it was the unit's first, second, or third mission during the rotation. Since several entity sets needed such an identifier, this process dominated the initial design phase.

An example of the entity set "Mission" is used to represent the typical entity set that was developed. Mission has for its Entity Identifier the predefined attribute "Mission Number." This allows the specific entity or mission that was completed at the NTC to be identified. Other descriptors which describe the mission but are common enough to prevent them from uniquely identifying the mission are included as the attributes of the entity. They include:

- "Mission Type" which describes whether the mission was an attack, defense, delay, etc.;
- the "Duration" which refers to the length of time that was used to complete the mission;

- the "Success" of the mission (a yes or no entry which would be a subjective evaluation by the observer/controller group);
- the "Location" which would describe a general location where the majority of the action took place.

As these attributes show, the entity gives a view of the particulars of the mission and nothing else.

Another entity set of interest is the entity set "Unit." This set is comprised of the individual units which attend the National Training Center as a rotational unit. In this set the identifier is "Unit Identification" and involves what the Army calls the UIC or Unit Identification Code. It is a six character code which reveals the units identity and extends down to identifying units at the battalion level. This attribute works smoothly as the Identifier since it incorporates a time proven method as the value. Other descriptors of the entity "Unit" which do not uniquely define the unit are:

- "Location" which is the home station of the unit;
- "Type" which identifies the unit as an armor or infantry unit;
- "Strength" which gives the total number of personnel in the unit;
- "Percent Filled - Officer" which represents the ratio of the number of officers currently available to the number of officers authorized;
- "Percent Filled - Enlisted" which represents a ratio of the number of enlisted personnel available to the number of enlisted personnel authorized.

As seen in "Mission," these attributes describe a Army unit in specific detail.

To reveal which unit was involved with a specific mission we must relate the entity set "Unit" to the entity set "Mission" to form a relationship. This relationship "Unit's Missions" reveals the applicable missions in which a unit was involved. Note that this is a 1:M relationship as a single unit's training at the National Training Center involves between 8 and 12 missions. Notice also that the relationship incorporate the definition of obligatory for the entity set "Mission" and non-obligatory for the entity set "Units." This stems from the fact that a mission can not exist unless it has been performed by a particular unit. The identifier precludes that a mission can be formed with only general values. On the other hand, a unit need not have attended the National Training Center yet to be included in the entity set "Unit." Therefore its membership within the relationship is not required in order to be present within the entity set. This relationship also contains no additional attributes which describe the match of the two particular entities. The diagram which represents this relationship is given in Figure 3.1.

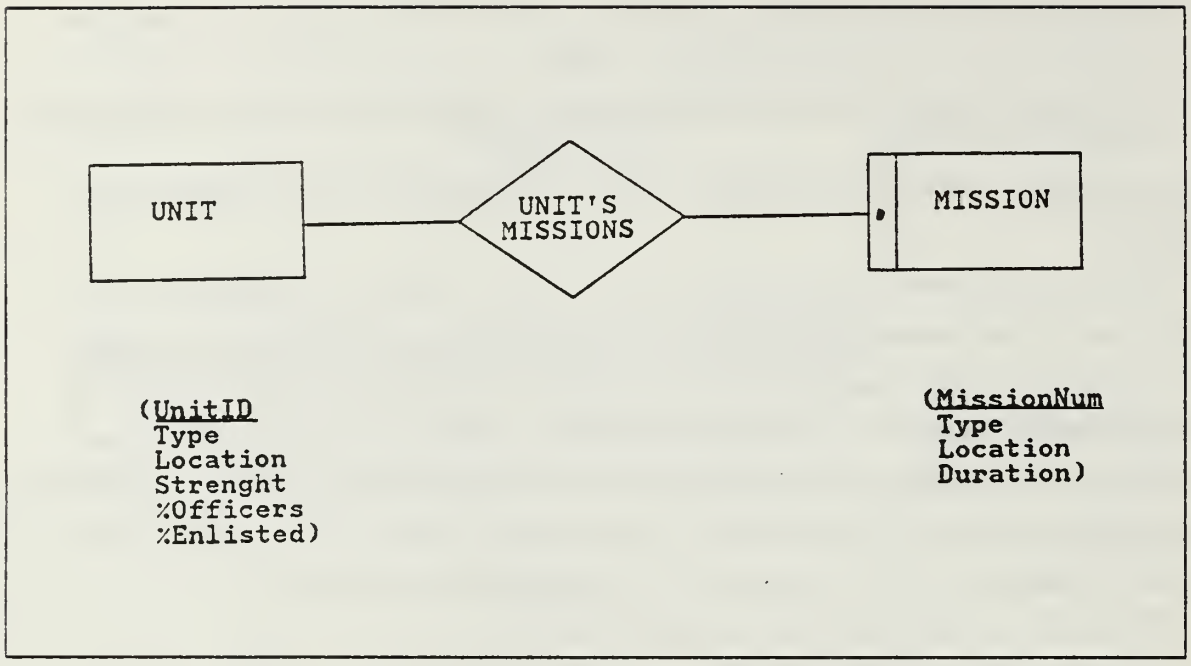


Figure 3.1 Unit's Mission Relationship.

Another relationship which involves a greater amount of work during the refinement process is Unit's Vehicle Use. This relation involves the entity set "Unit" which has been previously described and a new entity set "Vehicle." Vehicle is made up of the following:

- "Vehicle Number", the Entity Identifier, which uniquely identifies the vehicle;
- "Model" which describes the type of vehicle;
- "Armament" which gives the different weapons organic to the vehicle;
- "Suspension" which tells if the vehicle used a track or wheel suspension system.

The relationship created by the two entity sets has the identifiers "Unit Identification" and "Vehicle Number." The other attributes which complete the relationship are:

- "Miles," which relates the number of miles the vehicle was driven during the exercises;
- "Hours" which reveals the number of hours the engine was used;
- "NMC days" which gives the number of days the vehicle was unavailable for use due to mechanical problems;
- "Number of Missions missed" which refers to the number of missions that the vehicle missed while it was out for mechanical problems.

This relationship is M:N because many vehicles are used by a unit, but a single vehicle can be used by more than one unit. Notice that the Unit portion of the relationship is

obligatory since every rotational unit must use vehicles from the NTC, but not all vehicles are used by a unit. This relationship requires a much greater refinement because of the different dependence relationships existing between data values within the relational tables. These interdependencies will become clearer as the tables are developed from the relationship.

Using the process outlined in the previous section, the refinement process can begin. Using the first ER diagram, "Unit's Missions," tables will be made to reflect the data contained in "Unit" and "Unit's Missions." Notice that a distinct table that refers to "Mission" is not created since the obligatory relationship allowed the information within "Mission" to be combined with the information in the relationship. The tables in this first stage are provided in Figure 3.2. Notice at this point the table corresponding to "Unit's Missions" has a tuple with multiple entries in its columns.

At this point the normalization process can provide each tuple with a single value in each of the respective columns. This is done by using a specific tuple to represent each mission that a unit performs. Although this creates a redundancy of UIC values, it accomplishes the first step of the process. The tables are now in first normal form. To move to second normal form, the non-key attributes must be dependent upon all of the keys. The non-key attributes of "Unit" all depend upon the key "Unit Identification," since each is determined by the value of the key "Unit Identification." In the case of the second table "Unit's Missions," the non-key attributes are not all dependent upon the entire key, "Unit Identification" and "Mission number." The non-key attributes, "Mission Type," "Duration," "Success," and "Location," are all dependent upon the value of "Mission Number" only. To place this table into second normal form the table must be divided into two tables. The first one consists of "Unit Identification" and "Mission Number," and will be referred to as "Unit's Missions." The next one, called "Mission" will contain "Mission Number" and those attributes described before as dependent only upon that key. Figure 3.3 represents this action.

The tables are now in second normal form. To move into third normal form, the tables must have transitive dependencies removed. In the case of "Unit's Missions," this step is academic since the key incorporates both attributes of the table. In "Unit," all non-key attributes are dependent only upon the value of the key. There exists no other column within the table that helps determine the value of any other column. This places the table in third normal form also. The last table to inspect is "Mission." In that table a similar situation as in "Unit" exists. Although some units will contend





Unit

Unit ID	Type	Location	Strength	% Officers	% Enlisted
WAN6A0	Armor	Ft. Knox	531	90	83

Unit's Missions

Unit ID	Mission #	Type	Duration	Location
WAN6A0	860201	ATK	12 Hrs	Whale
	860202	DEF	24 Hrs	Whiskey
	860203	CATK	8 Hrs	Valley

Figure 3.2 First Step - relational model for 'Unit's Missions'.

that the location of a mission inadvertently determined the success of the mission, the attribute will be defined in a more rigorous way. Thus all tables are in third normal form.

The last step is to insure that the tables are in BCNF. The process will conclude at this point because further analysis and decomposition of the tables will create a large number of small tables. For the user needs, the gains of these actions are marginal at best. To reach BCNF, the values in each table which are a determinant for others, must all be key attributes. Again, in the case of "Unit's Missions," this is

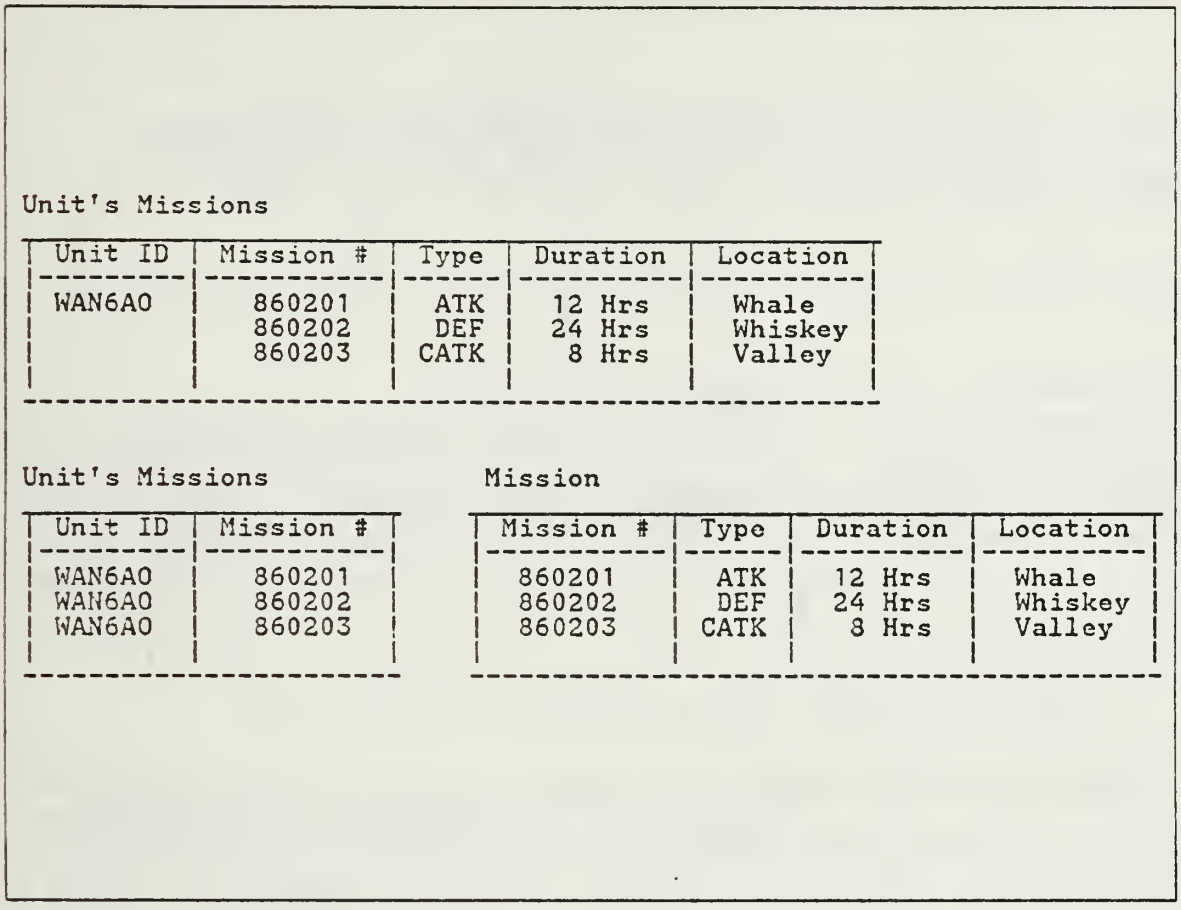


Figure 3.3 Second Step - relational model for 'Unit's Missions'.

academic since all attributes are key. In the tables "Unit" and "Mission," the values which determine the values of any other attribute within a specific tuple are the key attributes. "Unit Identification" determines the value of "location," "type," Strength," etc. and in a similar fashion the value of "Mission Number" determines the values of its respective tuples. At this point the normalization process for these three tables is complete and they are ready for the implementation phase. Their final forms are shown in Figure 3.4.

To refine the second ER diagram, "Unit's Vehicle Use," the same process is initiated. In an effort to highlight those areas of interest it is assumed that the reader has a firm understanding of how to arrive at the first normal form. At this point the tables reflect single values in all tuple column entries. Only two tables initially have been defined in this situation, because of the obligatory relationship with the entity set "Unit." Figure 3.5 reflects these tables.

Unit

Unit ID	Type	Location	Strength	% Officers	% Enlisted
WAN6A0	Armor	Ft. Knox	531	90	83

Unit's Missions

Unit ID	Mission #
WAN6A0	860201
WAN6A0	860202
WAN6A0	860203

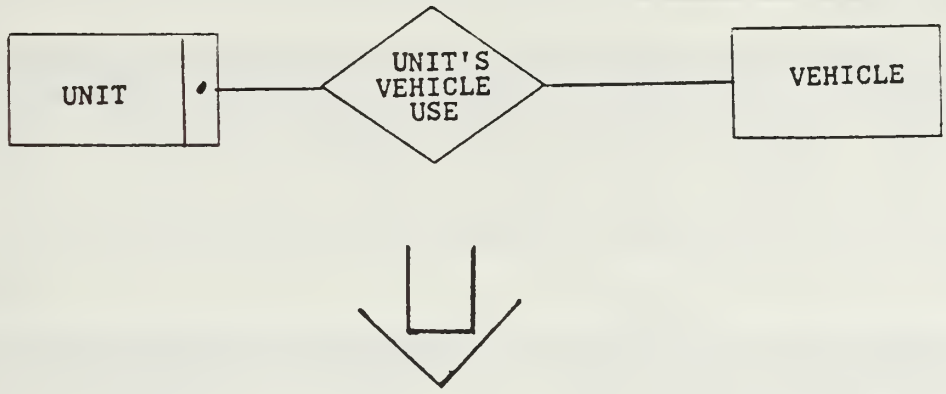
Mission

Mission #	Type	Duration	Location
860201	ATK	12 Hrs	Whale
860202	DEF	24 Hrs	Whiskey
860203	CATK	8 Hrs	Valley

Figure 3.4 Final Form - relational models for 'Unit's Missions'.

To place the tables into second normal form, the non-key attributes must be dependent upon the entire key of the table. In "Unit's Vehicle Use" the attributes of "Location," "Strength," "Type," etc. only depend upon "Unit Identification," so a similar table division process takes place as that which was used in "Unit's Missions." Continued refinement of "Unit" is academic since this was accomplished in the previous refinement process, so the discussion will be confined to "Unit Vehicle Use" and "Vehicle." By completing the division of "Unit" and "Unit Vehicle Use," all tables have been placed into second normal form. "Vehicle" is also in second normal form since all values are directly determined from the value within "Vehicle Number." In each instance the value of the non-keys is dependent upon the entire key. See Figure 3.6 for further clarification.

To move the tables into third normal form, the transitive dependencies must be removed. In "Unit's Vehicle Use" the values of "Miles," "Hours," and "NMC days" are all dependent upon the key since they reflect how a specific unit uses their vehicles in a particular rotation. However, the values for "Number of Missions missed" are dependent on the key and "NMC days." This is because the number of days a vehicle was unavailable for use also determines the number of missions that the vehicle is unable to participate in. To alleviate this problem another table is created which has



Unit's Vehicle Use

Unit ID I D	Veh #	Location	Strgth	%Offs	%Enl	Miles	Hours	NMC Days	Missions Missed
WAN6A0	E1075	Ft. Knox	531	90	83	2201	831	3	2

Vehicle

Vehicle #	Model	Armament	Suspension
E1073	M60A1	M68 CAN	Track
E1073	M60A1	M2 MG	Track
E1073	M60A1	M240 MG	Track

Figure 3.5 First Step - relational models for 'Unit's Vehicle Use'.

the same key but includes only the "Number of Missions missed." The values for "NMC days" are left out to reduce the redundancies within the tables since the key can adequately be used to separate this data. This new table will be refered to as "Missions missed." In the other table a different situation arises. Since "Model" determines "Armament," that table can be separated into two by moving "Armament" into a separate table with "Model" as the key. The column "Model" is maintained within the table "Vehicle" to allow the user to cross-reference and return a vehicle's organic weapons. This table will be called "Weapons."

The last step is placing the tables into BCNF. There are now four tables with which to work. "Unit's Vehicle Use" meets the criteria since all determinants are keys.

Unit's Vehicle Use

Unit ID I D	Veh #	Location	Strgth	%Offs	%Enl	Miles	Hours	NMC Days	Missions Missed
WAN6A0	E1075	Ft. Knox	531	90	83	2201	831	3	2

Unit

Unit ID	Type	Location	Strength	% Officers	% Enlisted
WAN6A0	Armor	Ft. Knox	531	90	83

Unit's Vehicle Use

Unit ID I D	Veh #	Miles	Hours	NMC Days	Missions Missed
WAN6A0	E1075	2201	831	3	2

Figure 3.6 Second Step - relational model for 'Unit's Vehicle Use'.

In "Missions Missed" the determinants are key, yet had "NMC days" been copied into this table, it would have been required to make "NMC Days" part of the key to fulfill the requirements of BCNF. In "Vehicle" all values in each tuple are determined by the value of the key "Vehicle Number." Finally, in "Weapons," the key "Model" determines the values of "Armament," so the process is complete.

At this point the tables have been processed in accordance with the procedure established in section B. All tables have been placed in Boyce-Codd Normal Form (BCNF), (see Figure 3.7). In places where an obvious redundancy was created, the process has been modified to create tables which are compact yet still able to process the user queries. The next section will describe the implementation process.

Unit

Unit ID	Type	Location	Strength	% Officers	% Enlisted
WAN6A0	Armor	Ft. Knox	531	90	83

Vehicle

Vehicle #	Model	Suspension
E1073	M60A1	Track

Weapons

Model	Armament
M60A1	M68 Cannon
M60A1	M2 MG
M60A1	M240 MG

Unit's Vehicle Use

Unit ID I D	Veh #	Miles	Hours	NMCs DAYS
WAN6A0	E1075	2201	831	3

Missions Missed

Unit ID	Veh #	Missions Missed
WAN6A0	E1075	2

Figure 3.7 Final Step - relational model for 'Unit's Vehicle Use'.

B. PHYSICAL IMPLEMENTATION OF THE PRODUCT

The relational model allows the smooth transition to an applications package. At the present time there are several database products which use the relational model as the basis for their implementation. Such products as *AppleWorks* and *DBase II* have incorporated the basic applications of the use of relational tables to provide the user with a working DBMS. In each product the user is presented with a language that closely approximates one of the languages spawned from relational calculus, namely the Transform Oriented language. Commercial designers have intentionally refined the application to be responsive to the needs of the average programmer with little or no background in relational theory.

Since the relational model consists of flat tables (two dimensional tables), it can be moved directly into a corresponding memory convention. The flat tables can be represented in memory as flat files of dimensions height (number of tuples) and width

(number of attributes within the tuple). This convention allows the tables to be conveniently moved into the memory source, be that either a floppy disk or a hard disk.

The issues of widespread use and portability of the information lead to the adoption of a commercially available product. In an effort to remain as responsive as possible, the product *DBase III* was chosen and later refined to use the its follow-on version *DBase III Plus*. A comparison of present capabilities show *DBase III Plus* to be the most useful application. This choice was also made based upon the perceived need of the user to work in a variety of locations independently. It also reflects an effort to make the transition to this new system as easy as possible for the user as the application utilizes a widely available microcomputer system.

The operations provided in *DBase III Plus* are a good logical base upon which to build. Although the language is one of the TO languages, and as such incorporates several relational calculus processes, it falls short of the needs of the user. This requires a revision of several of the facilities and the design of several additional options to fulfill the void. Within each new option, a series of additional error checking algorithms are implemented to ensure proper completion of each task.

The overall concept is to implement a system which is extremely user friendly, yet retains a great deal of power and flexiblity. This is accomplished by the use of a menu driven atmosphere from which the user can perform his work. Each task is provided the ability to cease execution and return to the start point, as a worst case scenario. Although the user is assumed to be an intelligent one, the system was none-the-less "gorilla proofed."

Along with the basic operations of creation, addition, deletion, and listing, the user is provided with several added features. These are broken into two groups: one for information control, the other for information manipulation. Information control is accomplished through the development of categories of information under which the appropriate types of information are stored. The user is provided a menu within each operation from which he is able to pick a category that best describes the data he seeks. This helps the user to better organize the different files of data into logical storage structures. The other aspect of information control is achieved through the implementation of specific operations to restore and save data that the user has changed or added. This option utilizes a secondary storage facility in both cases.

In terms of information manipulation, the language provided within *DBase III Plus* was tailored to provide additional capabilities for the user. These changes are added to move the language closer to approximating the capabilities found in relational calculus languages. With the user's needs as a guide, it was decided that he required the ability to be able to project specific columns of values from within a table, select specific tuples within a table, join two or more tables together upon common attributes, identify membership within a table, and accomplish some basic analytical functions on the data within the tables. With the exception of the mathematical analysis functions, these operations in reality closely approximate the relational calculus join, project, and select operations which are embedded within that language.

These operations allow the user a much greater degree of flexibilty and reflect an engineering viewpoint with the user's needs as the foremost consideration. These additional capabilities of the language require only that the user to know a general location of the data and what kind of value is to be returned. A more specific description of each function is provided within the user's manual chapter.

In implementing the relational model, the following steps are taken. First, all relational tables are divided into the categories which best describe the information stored in them. Each category is implemented as a file of one column. That column contains a listing of the tables which make up that category. Next, each relational table is then implemented as a file which contains columns reflecting the attributes of the table. Once the tables are loaded into memory files, the language implementation can be initiated.

The final step is the implementation of the DBMS program to incorporate the use of menus to move from step to step and the addition of the required operations. These menus are designed to reflect an average understanding of the terms used to describe DBMS operations. In the instances where the understanding might be suspect, an explanation is provided. The incorporation of the added operations is done by the use of a procedure file. The procedure file is made up of a series of smaller program files, each accomplishing a specific task. The file is loaded at the start of the session and remains throughout. At the conclusion of the session, the user is presented the option to save his work. Throughout the session, at no time is the user required to make an entry that is not presented on the monitor screen.

The code of the various options is provided as an appendix. To highlight the added features the following descriptions of the operations are provided. The Join

operation is built upon the provided option and basically ensures that the operation exists only when a common attribute is available. The Select operation allows the user to select specific attributes within a given table and save them in another file of his choice. The Find operation allows the user to locate an entry anywhere within a given table and save those in a file of his choice. The user is only required to know a table location of the data, as each column value is checked automatically. Membership is also confirmed or denied using the Find operation. The Analytical operations provide the user with a mathematical analysis of the entries of a column within a table. Such calculations as the mean, standard deviation, and others are returned. As a final feature, the user is provided with a tutorial section which walks him through each different option, as one of the initial menu choices.

IV. RECOMMENDATIONS FOR IMPROVEMENT OF NTC DATABASE

A. CHAPTER CONCEPT

The intent of this chapter is to identify areas which might benefit from further modification or system integration. These areas can be divided into three distinct groups. The first area, Program Code, identifies aspects of the code which if modified, would create a more responsive and efficient product. The next area, Systems Integration, highlights outside influences that could be incorporated into the product to benefit the system as a whole. The last area of concern, National Training Center Aspects, reveals areas of data collection that would increase the practical benefits of the information within the database. These aspects represent areas which would significantly contribute to the product accomplishing its primary mission of analyst support.

B. PROGRAM CODE MODIFICATIONS

The program code can be modified in several areas to maximize its efficiency and responsiveness. Program code modifications would raise the overall program efficiency by allowing the program to maximize the amount of operations possible within a given amount of time. The first area which would benefit from code modification is the search algorithm used to locate an object within the "Find" operation. Presently the program uses a sequential search routine. A more efficient algorithm that would perform a "smart" search and eliminate redundant or unnecessary values would allow the user to receive his answer in a shorter amount of time. However, the new search algorithm should not compromise the quest for speed by requiring the user to enter any additional information other than what is presently required. The other aspect of the code that lends itself to modification is the overall modularity of the program. Presently the code has been developed to show system feasibility and applicability to a specific use. Should the user wish to utilize the program over an extended period of time, portions of the code would be easier to maintain if they were rewritten to be more modular in composition.

The system responsiveness reflects the program's ability to satisfy the needs of the user and could be upgraded by incorporating three additional features. The first would be the option to allow the user to exit the program, yet remain within the DBase environment. This would allow the user a greater degree of flexibility during a given work session. The next feature would require an observation of the analyst's program use over time. The mathematical operations could be modified to reflect those areas of greatest need, thus increasing that choice's responsiveness to the needs of the user. As these needs could fluctuate with time, the most appropriate means to accomplish this might be to include an additional operation choice within the mathematical operation menu. The last area which might benefit the user is the incorporation of mouse functions. These functions allow easier user interface with the program, and since they can be mastered quickly, could significantly reduce user training time.

C. SYSTEMS INTEGRATION ISSUES

Integration issues impact significantly upon the system's overall ability to handle and process reliable data. Changes within this area reflect how the program interacts with the outside world (including the machine environment). These modifications reflect the need to allow a more efficient means of receiving input and distributing output. Input could be more efficiently collected by the implementation of a interface program to automatically load the data tables from the local workstation (reference Appendix J, SAIC contract upgrade of home station facilities). This would remove part of the manual input process and allow the system to be more responsive to local data. It would also allow the system to be less dependant upon unit feedback reports to generate the data tables. In addition to adding the workstation to unit home stations, SAIC is also upgrading the collection facilities at the NTC. Changes such as the addition of air assets into the engagement system and the addition of an elevation axis will allow the data to be more realistic. This will require an update to the tables to allow them to reflect this additional information.

Output needs can be solved by the incorporation of a means to print the screen displays. The overriding concern of this implementation is unit confidentiality. Therefore within the "print" mode should be the incorporation of a means to allow units to remain "hidden." The only information displayed for all should be the data values which represent answers to the queries, excluding unit identification.

D. NATIONAL TRAINING CENTER RECOMMENDATIONS

The last area of interest concerns the basic collection methods that the National Training Center has established as its standard operating procedures. The inclusion of several areas of data into the overall evaluation package would allow for a more comprehensive evaluation. Specifically, the inclusion of a unit history profile would allow a direct evaluation to be made of the unit's performance based upon past exposure. This alone could allow researchers and analysts to truly gauge the effectiveness of the NTC experience. The incorporation of logistical data could be used to validate a combat Prescibed Load List (PLL) that would allow units to effectively operate in similar terrain. The inclusion of recorded medical operations could more effectively evaluate the Army's present field evacuation doctrine and update it as needed. A more inclusive unit report could help validate the necessity of including the many intangible aspects of combat actions, which to date have been passed on from unit to unit rather than formalized. Practices such as long distance boresighting and effective sleep plans could be given a solid evaluation rather than the heresay evidence approach. These aspects all revolve around the inclusion of a wider range of subjects into the evaluation package in an effort to make the total experience more profitable. Although these additions will not guarantee better individual performance predictors, they will allow much more thorough trend analysis to occur. Appendix I includes a possible means to collect this data through specific evaluation sheets.

In terms of a relation to this product, those incorporated areas could produce a wider range of data tables from which the user could draw (after the system was modified to reflect them). This would allow for a better analysis to be conducted since all significant aspects of the event could be related together. By factoring in all possible input, the user could generate a truer appraisal of the event, thereby ensuring a more responsive answer to his query.

APPENDIX A
ENTITY-RELATIONSHIP DIAGRAMS

The following diagrams are provided to the user to reveal how the real world situation was modelled. Each entity contains the respective attributes in parentheses and the key attribute is underlined. The keys for the relationships are understood to be a combination of the keys of the respective entities, and as such only the relationship attributes are presented.



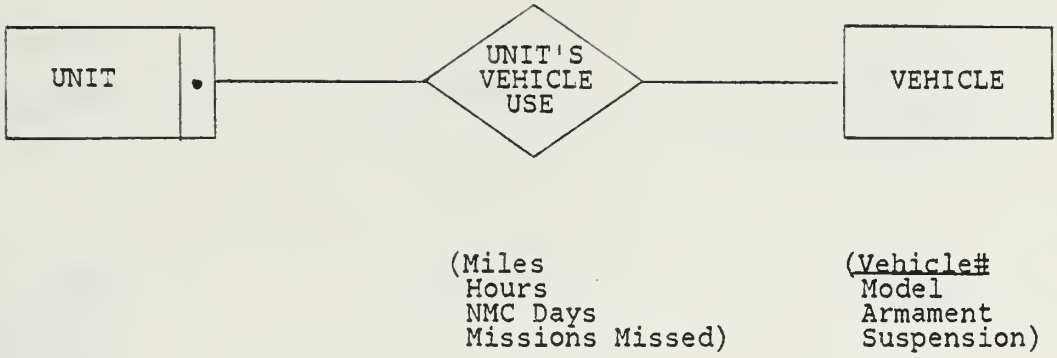
(MissionNum
Type
Duration
Location)

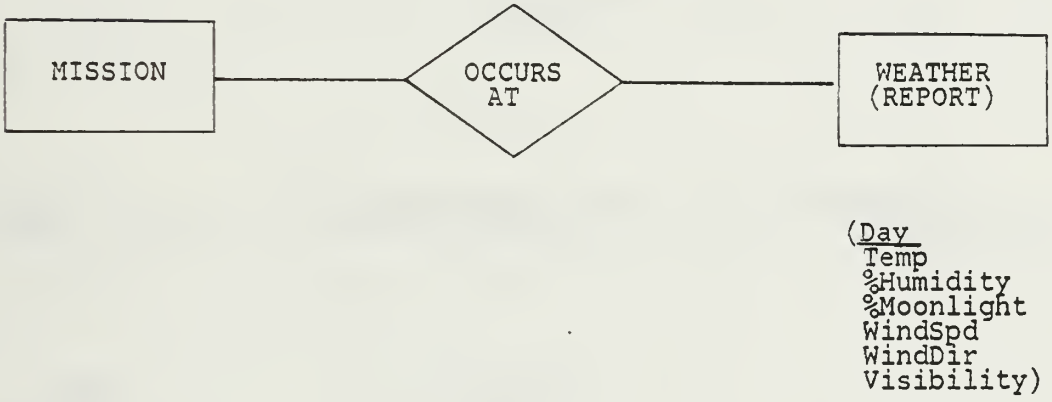
(TotalFired
Total4.2
Total155
Total8in
TotalFrat)

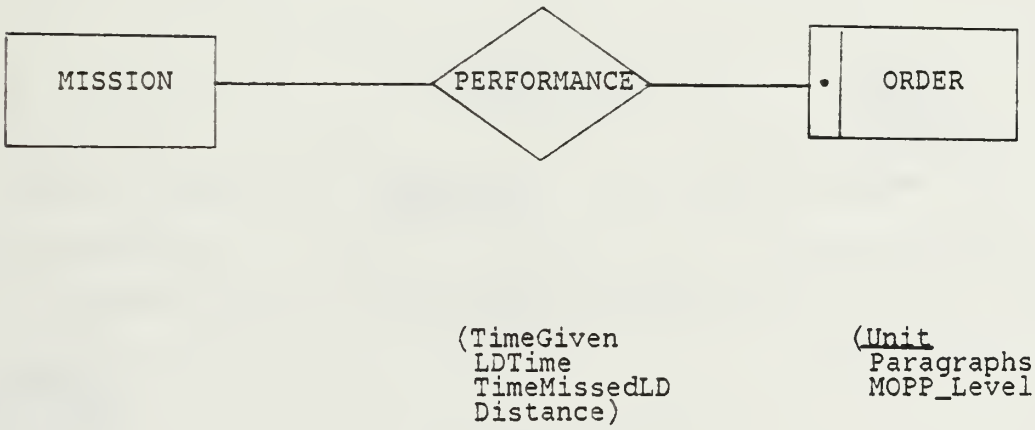
(TargetNum
Num_rds
Num_tubes
ProcessTime
Hit/Miss
Type_Rnd)



(UnitID
Type
Location
Strenght
%Officers
%Enlisted)


UNIT
UNIT'S
VEHICLE
USE
VEHICLE
(Miles
Hours
NMC Days
Missions Missed)
(Vehicle#
Model
Armament
Suspension)


MISSION
OCCURS
AT
WEATHER
(REPORT)
(Day
Temp
%Humidity
%Moonlight
WindSpd
WindDir
Visibility)


MISSION
PERFORMANCE
ORDER
(TimeGiven
LDTime
TimeMissedLD
Distance)
(Unit
Paragraphs
MOPP_Level)



(Unit
Boresight
Distance
Tac_Feed
Sleep_plan
Tac_Fuel
%Security)



(Amount
Rte_Distance
Trip_Time)

(Class
Item
Sch/Emergency)



(PartNum
PartName
NMC_Status
Major_Assy)



(Qty_Used)

(CLass Name)



(ReasonCode
Dn_Time)

(Model#
Number_Start
Number_Failed)

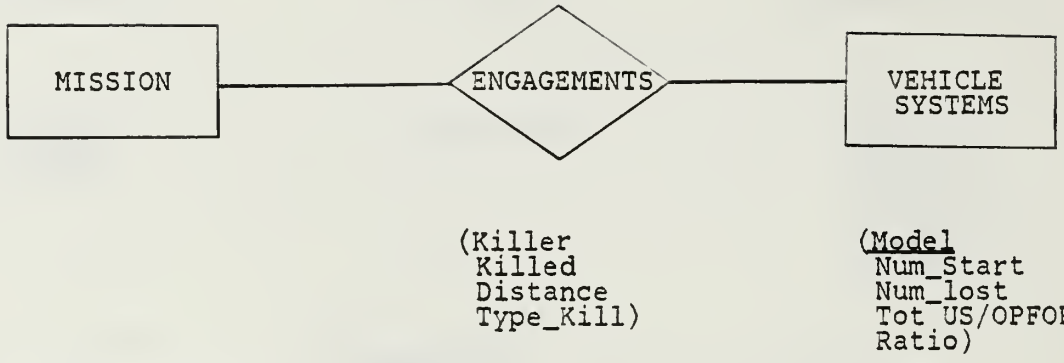


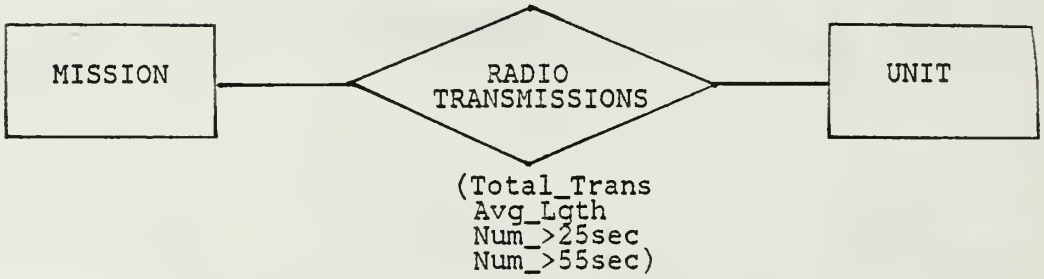
(Aid_Stat
Field_Hosp)

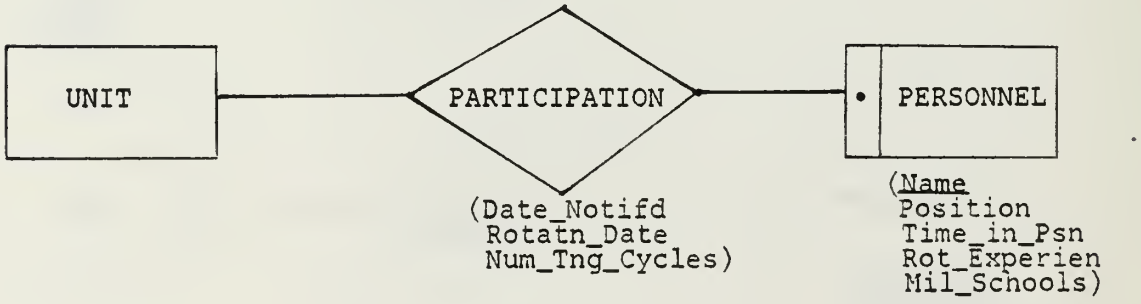
(Unit
Type
KeyLdr
Number)

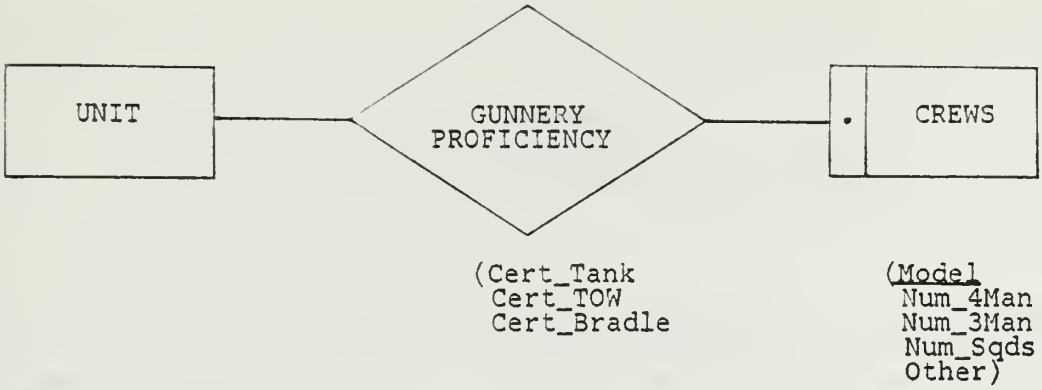


(NBC_Agent
Pre_Warn
MaskTIme
Agent_Rptd
Num_Caslts
Area_Prediction)


MISSION
ENGAGEMENTS
VEHICLE SYSTEMS
(Killer
Killed
Distance
Type_Kill)
(Model
Num_Start
Num_lost
Tot US/OPFOR
Ratio)


MISSION
RADIO TRANSMISSIONS
UNIT
(Total_Trans
Avg_Lgth
Num_>25sec
Num_>55sec)


UNIT
PARTICIPATION
PERSONNEL
(Date_Notifd
Rotatn_Date
Num_Tng_Cycles)
(Name
Position
Time_in_Psn
Rot_Experien
Mil_Schools)


UNIT
GUNNERY
PROFICIENCY
CREWS
(Cert_Tank
Cert_TOW
Cert_Bradle
(Model
Num_4Man
Num_3Man
Num_Sqds
Other)

APPENDIX B
DATABASE TABLES

1. INTRODUCTION AND DEFINITION OF FORMAT

The following appendix presents to the user the various tables which are used within the database. Each table has been through the respective normalization process discussed within the preceding chapters of the thesis, and are presently in Boyce-Codd Normal Form (BCNF). The tables are presented within their respective category and are formatted in the following manner: The first column given is the field name, the second is a description of the field, and the third is a location where the data can be found. This is in compliance with the thesis contract with FT. Knox. The name of the table appears at the top of these columns along with the respective name used within the database.

The data locations are presented in a coded form to allow for a more easily understood format. Combinations of code reflect the need to combine the values of both locations to arrive at the correct input for this table. The code is explained within Figure B.1

2. DATA TABLES

A. Fire Support Tables

Fire Support Targets (FSTARGET)

MissionNum	-	The number of the tactical mission	-	6 / 1:18
TargetNum	-	The number of the target fired	-	1:5
NumberRds	-	The number of rounds fired	-	1:7
NumberTubes	-	The number of firing gun tubes that fired at target	-	1:15
ProcessTime	-	The time required to complete the fire request	-	1:15
Hit-Miss	-	Whether the target was hit or missed	-	1:25
TypeRound	-	What type of artillery unit was used in the mission	-	1:15

Fire Support Totals (FSTOTAL)

MissionNum	-	The number of the tactical mission	-	6 / 1:18
TotalMissions	-	The number of fire missions for this tactical mission	-	5
Total4.2	-	The number that used 4.2" mortars	-	5
Total155	-	The number that used 155mm artillery batteries	-	5
Total8In	-	The number that used 8 inch artillery batteries	-	5

1 - Data is located within present collection system's tape of a rotational unit's performance. A respective ingres table number is also given to further identify data. (e.g. - 1:18 refers to ingres table #18 within the reference) .ref 10 . ' p. 2-38'

2 - Data is within combat vehicle contractor's database

3 - Data is contained in post DS4 database, which tracks logistical requisitions.

4 - Data is contained within a respective account which the unit maintains to meet rotational exercise requirements/needs.

5 - Data would be the result of the use of one the additional checklists provided in Appendix I.

6 - Data requires new assignment generation techniques, (i.e. mission number would be the combination of the home station-calendar year-number of mission).

Figure B.1 Data Location Coding Formats.

Fire Support Fratricides (FSFRAT)

MissionNum	-	The number of the tactical mission	-	6 / 1:18
NumFratricides	-	The number of fire missions that resulted in friendly casualities by artillery round explosions	-	5

B. Intangibles

Company Boresights (INBORE)

MissionNum	-	The number of the tactical mission	-	6 / 1:18
CompanyTeam	-	The identity of the unit reported	-	1:15
Boresight	-	Whether or not the unit boresighted weapons (Y/N)	-	5
AverageDist	-	The average distance at which the boresight was done	-	5

Tactical Resupply Actions (INRESUP)

MissionNum	-	The number of the tactical mission	-	6 / 1:18
CompanyTeam	-	The identity of the unit reported	-	1:15
TacticalFeed	-	Whether or not the unit fed tactically (Y/N)	-	5
SleepPlan	-	Whether or not the unit used a sleep plan (Y/N)	-	5
PercentSec	-	The percentage of personnel on security during mission	-	5
TacticalFuel	-	Whether or not the unit refueled tactically (Y/N)	-	5

Brigade to Battalion Order Time (INORDER1)

MissionNum	-	The number of the tactical mission	-	6 / 1:18
BdeOrderTime	-	Time at which the brigade gave its order	-	5
BnOrderTime	-	Time at which the battalion gave its order	-	5
LDTime	-	Time at which the unit crossed the Line of Departure	-	5

Company Order Times (INORDER2)

MissionNum	-	The number of the tactical mission	-	6 / 1:18
CompanyTeam	-	The identity of the unit reported	-	1:15

CoOrderTime	-	Time at which company gave its order	-	5
Platoon	-	Platoon designator	-	5
PltOrderTime	-	Time at which that platoon gave its order	-	5

Company MOPP Level (INMOPP)

MissionNum	-	The number of the tactical mission	-	6 / 1:18
CompanyTeam	-	The identity of the unit reported	-	1:15
MOPPLevel	-	Level of MOPP used during the mission	-	5

Company Order Evaluation (INORDER3)

MissionNum	-	The number of the tactical mission	-	6 / 1:18
CompanyTeam	-	The identity of the unit reported	-	1:15
OrderPara	-	paragraph in OPORD critique (1 - 5)	-	5

Weather Statistics (INWETH)

MissionNum	-	The number of the tactical mission	-	6 / 1:18
Temp	-	Temperature at LD Time	-	5
PercentHumid	-	Percent humidity at LD Time	-	5
PercentMoon	-	If Night mission, percent moonlight available	-	5
WindSpeed	-	Wind Speed for the mission	-	5
WindDirection	-	Direction that wind was blowing during mission	-	5
Visibility	-	Visibilty distance during the mission	-	5

C. Logistical

Vehicle (LGVEH)

VehicleNum	-	Identification number of the vehicle	-	2
Model	-	Type vehicle assigned	-	2
Suspension	-	Type of suspension on vehicle(track/wheel)	-	2

Unit's Vehicle Use (LGUNVEH)

VehicleNum	-	Identification number of the vehicle	-	2
Model	-	Type vehicle assigned	-	2
Miles	-	Number of miles used during the rotation	-	2
Hours	-	Number of hours recorded on the rotation	-	2
Rounds	-	If applicable, number of rounds fired thru gun	-	2
NMCDays	-	Number of Non Mission Capable days	-	2

Weapons (LGWEAPON)

Model	-	Type of vehicle	-	2
Armament	-	Weapons which are organic to model	-	2

Mission Missed (LGMISSED)

Unit ID	-	Identification of Unit Using vehicle	-	2
VehicleNum	-	Identification number of the vehicle	-	2
MissionMissed	-	Number of tactical missions missed by vehicle	-	2

Vehicle Parts History (LGPARTS)

VehicleNum	-	Identification number of the vehicle	-	2
PartNumber	-	Part number of part that vehicle needed	-	3
PartDescrip	_	Short part name identifying part	-	3
NMCPart	_	(Y/N) did this part deadline the vehicle	-	2

Unit Fuel History (LGFUEL)

UnitIden	-	Identification of the tactical unit	-	4
Diesel	-	Gallons of diesel used by unit	-	4
GalTrucks	-	Number of gallons used by trucks	-	5
GalTanks	-	Number of gallons used by tanks	-	5
GalAPC	-	Number of gallons used by armored personnel carriers	-	5
GalBrad	-	Number of gallons used by Bradley	-	5

		carriers		
GalTows	-	Number of gallons used by TOW vehicles	-	5
		Unit Mileage Figures (LGMPG)		
UnitIden	-	Identification of the tactical unit	-	4
MPGTanks	-	Miles per gallon figure for tanks	-	6
MPGTrucks	-	Miles per gallon figure for trucks	-	6
MPGAPC	-	Miles per gallon figure for armored personnel carriers	-	6
MPGTOW	-	Miles per gallon figure for TOW vehicles	-	6
		Water and Ration Use (LGRATION)		
UnitIden	-	Identification of the tactical unit	-	4
GalsWater	-	Number of gallons of water used by unit	-	4
NumMRE	-	Number of Meals Ready to Eat consumed by unit	-	4
		Vehicle Statistical History (LGSTATS)		
MissionNum	-	Number of the tactical mission	-	6 / 1:18
NumTanksStart	-	Number of tanks at the start of the mission	-	1:2
NumTanksDrop	_	Number of tanks lost for maintenance reasons	-	1:15
NumAPCStart	-	Number of personel carriers at the start of the mission	-	1:2
NumAPCSDrop	_	Number of personel carriers lost for maintenance reasons	-	1:15
NumBradsStart	-	Number of Bradley vehicles at the start of the mission	-	1:2
NumBradsDrop	_	Number of Bradley vehicles lost for maintenance reasons	-	1:15
NumTOWsStart	-	Number of TOW vehicles at the start of the mission	-	1:2
NumTOWsDrop	_	Number of TOW vehicles lost for maintenance reasons	-	1:15
		Vehicle Deadline Status (LGDEAD		
MissionNum	-	Number of the tactical mission	-	6 / 1:18
DnVehType	-	Type of vehical that was deadlined	-	2
ReasonCode	-	Coded entry showing reason vehicle was deadlined	-	2
DnTime	-	Number of hours that the vehicle was deadlined	-	2
		Major Assembly Usage (LGASSY)		
UnitIden	-	Unit identification number	-	3
NumMajorAssy	-	Total number of major assemblies used during rotation	-	3
NumTank	-	Number of tank major assemblies	-	3
NumTOW	-	Number of TOW major assemblies	-	3
NumAPC	-	Number of APC major assemblies	-	3
NumTruck	-	Number of truck major assemblies	-	3
NumBrad	-	Number of Bradley major assemblies	-	3
		Unit Supply Requests (LGREQST)		
MissionNum	-	Number of the tactical mission	-	6 / 1:18
Unit	-	Unit identification	-	3 / 4
SupClass	-	Class of supply requested	-	3 / 4
Item	-	Item description	-	3 / 4
Amount	-	Amount of the item requested	-	3 / 4
Sch/Emerg	-	Whether the request was scheduled refill or emergency	-	5
		Unit Supply Routes (LGROUTE)		
MissionNum	-	Number of the tactical Mission	-	6 / 1:18
Unit	-	Unit identification	-	5

Distance	-	Distance from tarveled by resupply vehicles	-	5
TripTime	-	Time needed to make the trip	-	5

Medical Evacuation Summary (LGMED)

MissionNum	-	Number of the tactical mission	-	6 / 1:18
NumCasualties	-	Number of casualties during mission	-	1:3
NumAidStation	-	Number taken to Bn Aid Station	-	5
NumFldHosp	-	Number evacuated to Field Hospital	-	5

D. Tactical

Mission Summary (TABATTLE)

MissionNum	-	Number of the tactical mission	-	6 / 1:18
Type	-	Type of mission conducted	-	1:15
Duration	-	Length of time that mission ran	-	1:18
Location	-	General area location of action	-	6

NBC Actions (TANBC1)

MissionNum	-	Number of the tactical mission	-	6 / 1:18
Unit	-	Unit identification	-	5
AgentType	-	Type agent encountered during mission	-	5
PreWarn	-	(Y/N) was unit prewarned of attack	-	5
TimeToWarn	-	Time unit took to get into MOPP 4	-	5
NumCasualties	-	Number of casualties as result of attack	-	5

NBC Messages (TANBC2)

MissionNum	-	Number of the tactical mission	-	6 / 1:18
Unit	-	Unit identification	-	5
AgentType	-	Type agent used by OPFOR on attack	-	5
AgentReported	-	Type agent reported by unit	-	5
AreaPred	-	(Y/N) was an area prediction calculated	-	5

Vehicle Kill Summary (TAKILLS)

MissionNum	-	Number of the tactical mission	-	6 / 1:18
System	-	Fighting vehicle model	-	1:15
NumStart	-	Number that atarted the mission	-	1:2
NumLost	-	Number killed during the mission	-	1:2
NumT72	-	Number killed by OPFOR tanks	-	1:2
NumBMP	-	Number killed by OPFOR BMPs	-	1:2
NumSagger	-	Number killed by OPFOR Sagger missiles	-	1:2
NumSP122	-	Number killed by OPFOR SP 122 howizters	-	1:2
NumRPG	-	Number killed by OPFOR RPG rockets	-	1:2
NumCAS	-	Number killed by OPFOR Close Air Support	-	1:2
NumARTY	-	Number killed by OPFOR Artillery	-	1:2
NumOther	-	Number killed by other means	-	1:2

Loss Ratio (TALOSS)

MissionNUm	-	Number of the tactical mission	-	6 / 1:18
VehicleType	-	Type vehicle losses	-	1:15
USLoss	-	Number of friendly vehicles lost	-	1:15 /1:2
OPFORLoss	-	Number of OPFOR vehicles lost	-	1:15 /1:2
Ratio	-	Kill ratio US/OPFOR	-	6

Key Leader Losses (TALDR)

MissionNum	-	Number of the tactical mission	-	6 / 1:18
Unit	-	Unit identification	-	1:4
Position	-	Position of the leader killed	-	1:3

Radio Transmission Summary (TARADIO)

MissionNum	-	Number of the tactical mission	-	6 / 1:18
Unit	-	Unit identification	-	1:15
NumTrans	-	Number of transmissions made	-	1:2
AvgLength	-	Average time length of the transmissions	-	1:3
NumGTR25	-	Number that were longer than 25 seconds	-	1:2
NumGTR55	-	Number that were longer than 55 seconds	-	1:2

Unit Line of Departure Statistic (TALOD)

MissionNum	-	Number of the tactical mission	-	6 / 1:18
Unit	-	Unit identification	-	1:15
LDOnTime	-	(Y/N) did unit cross LD on Time	-	5
Time	-	Amount of time unit missed LD time by	-	5
Distance	-	Distance from assembly area to LD	-	5

Direct Fire Targets (TADFTGT)

MissionNum	-	Number of the tactical mission	-	6 / 1:18
KillerVeh	-	Identification of killer vehicle	-	1:3
KilledVeh	-	Identification of vehicle which was killed	-	1:3
WeaponSys	-	Weapon System used to kill with	-	1:22
Distance	-	Distance between vehicles - Range of shot	-	1:10

E. Unit History

Unit Summary (TAUNIT)

UnitIden	-	Unit Identification	-	5
Type	-	Type of unit (ARMor/ INFantry)	-	5
Location	-	Home Station of unit	-	5
Strength	-	Personnel assigned at rotation time	-	5
PercenOff	-	% officers available / officers required	-	5
PercenEnl	-	% enlisted available / enlisted required	-	5

Unit's Missions (UHMISSNS)

UnitIden	-	Unit identification	-	5
MissionNum	-	Number of the tactical mission	-	6 / 1:18

Unit Personnel Summary (UHPER)

UnitIden	-	Unit identification	-	5
Position	-	Position description	-	5
TimePsn	-	Length of time that individual has filled that position	-	5
NumRotations	-	Number of previous rotations experienced	-	5
MilSchools	-	Military schools	-	5
PrevRotPsn	-	Previous positions held during other rotations	-	5

Unit Training Information (UHTNG)

UnitIden	-	Unit identification	-	5
DateNotified	-	Date that unit was notified of upcomming rotation	-	5
RotationDate	-	Date of rotation	-	5
NumTngCycles	-	Number of training cycles in between dates	-	5

Unit Crew Status (UHCREW)

UnitIden	-	Unit identification	-	5
Num4ManCrews	-	Number of 4 man crews on tanks	-	5
Num3ManCrews	-	Number of 3 man crews on tanks	-	5
NumFullSqds	-	Number of filled squads in unit	-	5
NumOtherSqds	-	Number of unfilled squads in unit	-	5

Unit Crew Certification (UHCERT)

UnitIden	-	Unit identification	-	5
CertCrewTanks	-	Number of tanks crews that are certified during rotation	-	5
CertCrewTOWs	-	Number of TOW crews that are certified during rotation	-	5
CertCrewBrad	-	Number of Bradley crews that are certified during rotation	-	5

APPENDIX C
USER HANDBOOK FOR DBMS PROGRAM

1. CONCEPT OF THE USER'S MANUAL

This chapter will provide a user's manual to the user or reader. The explicit purpose of this manual is to allow the user to operate the database program in an intelligent manner. This section will provide three distinct sections for the user that will allow the user to implement the system properly. The only assumptions made at the start of this chapter are that the user is working on an IBM AT microcomputer (or a similar system), that the user has a working knowledge of the basic DOS functions provided by the operating system, and that the user has a working copy of *DBase III Plus* on his system.

The three major sections contained within the chapter are first, the section designed to properly implement the system from the floppy disk, second, a general discussion of the menu options contained within the system, and last, suggestions on how to maintain a logical working system. The first section will take the user through a step by step "boot up" session to ensure that the system is properly loaded into his microcomputer. The second section will show through an example session. the various choices presented to the user. The last section will provide some insights and suggestions that will ensure that the logical organization of the data remains intact. It will also provide some useful hints to help maintain the system's effectiveness.

The applications program follows a menu driven format. The user is presented with a variety of choices within each menu and the abiility to "quit" that menu and return to a previous one. Normally in any situation the user can exit that option by typing the letter "q", with the exception of certain categorizing choices that force the user to select an option. These are conducted to maintain the proper functioning of the program. Should the user be displeased with this action he can always delete the table he has just created using the appropriate choices. In no situation is the user presented with an option where his choice is anything other than what is displayed upon the monitor.

2. LOADING THE APPLICATION PROGRAM

This section will provide the user with a step by step discussion of the proper procedure used to load his program. At this point, the discussion will assume that the user has turned his machine on, booted the system with an operating system such as DOS 3.1 and the prompt shown in Figure C.1 is displayed on his monitor screen.

```
C>
```

Figure C.1 Starting Point for Upload Procedure.

If this is not the case, please consult the microcomputer's user manual to move the system to this display.

Once the user has reached this display, he is ready to proceed. The first action should be to move to the sub-directory which contains the *DBase III Plus* program. This can be accomplished through the command (after the prompt) "cd < sub-directory name>". When this has been completed the user should take the application's floppy disk and place it into the "B" disk drive. Then the user changes drives to the "B" drive. Figure C.2 gives a listing of the commands used to date.

```
C>cd <sub-directory name>
:
C>B:
:
B>
```

Figure C.2 Commands entered to reach B drive.

At this point the user should enter the following command after the "B" drive prompt: "copy startup.bat C:". This will load the command file "startup" into the proper sub-directory. When the file has been loaded the user returns to the sub-directory by entering "C:" after the "B>" drive prompt. The user now enters the command "startup" after the "C" drive prompt and waits for the completion of the load process. Although the user could have directly transfered all files to the "C" drive, this

procedure ensures that only the proper starting files have been passed. The disk in the "B" drive now becomes the system backup and this procedure can be done any time in the event of a system failure or a move to another microcomputer. The user can now remove the floppy disk from the "B" disk drive, but it is recommended that for the duration of the session, he keeps it in place. Figure C.3 reviews the entire load procedure.

```
C>cd <sub-directory name>
C>B:
B>
B>copy startup.bat C:
B>C:
C>startup
```

Figure C.3 System Upload Commands.

At this point, the user is now able to properly call and execute the program. By following the above procedure, he is ensured that all applicable components of the application have been properly stored.

3. PROGRAM CHOICES AND CAPABILITIES

The application program which has been provided to the user is a powerful vehicle that has been designed for the data analyst. The program provides the user with the ability to locate and analyze data on a variety of subjects. The different operations allow the user to co-locate data, break tables into specific columns of data, and work through queries to arrive at meaningful output. Inherent in this analysis, is the requirement to provide a cursory analysis of a data column. This capability is specifically designed into one of the option choices.

The user begins implementation of the program by typing "NTC" at the operating system prompt (C>). This executes the application program implementing *DBase III Plus*. The user is then presented with the copyright disclosure statement and

then a "welcome" screen (see Figure C.4). After the user types any key, the procedure file is loaded and the top menu is provided. The user is presented a tree type structure in Figure C.5 which will facilitate understanding the menu interactions. Upon starting the program, the user is presented with the top menu where five choices are presented. Selecting any choice (other than "quit") places him in the next level where he can follow the tutorial or choose an option from another menu. These menu choices all implement an action that manipulates a data file from the database. Once the required actions have been completed, the user can traverse back "up" the tree by choosing the "quit" option in a menu display. Ultimately this will allow the user to exit the program.



Figure C.4 Welcome Statement.

a. **Initial Menu of Program**

The first or "top" menu that is displayed for the user presents him with five choices (see Figure C.6). The first choice is to invoke the text file "Tutorial" and read a prepared text file that outlines the program in a user manual style. This is provided to allow immediate support to the user while he is operating the program. The next three choices move the user user to a second menu level and present the user with choices which accomplish specific actions. The last choice from the first menu is the "quit"command. This choice is invoked when the user wishes to leave the program gracefully. Prior to actually moving back to the control of the operating system, the user is given the opportunity to do file maintenance in order to delete unnecessary files,

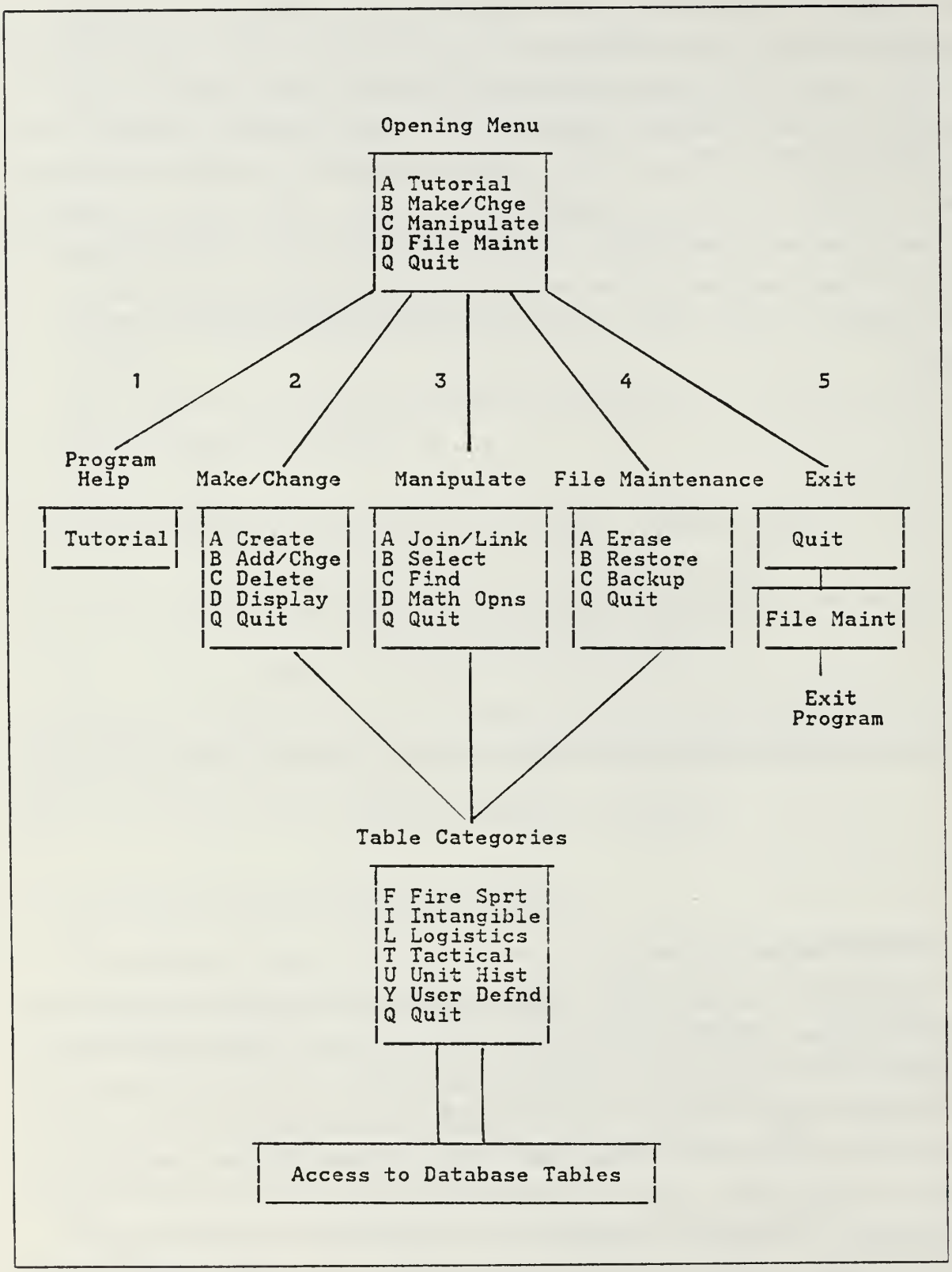


Figure C.5 Menu Interaction Tree.

restore a lost files, or save files that may have been changed during the session. In order to pass this phase, the user must make a correct entry.

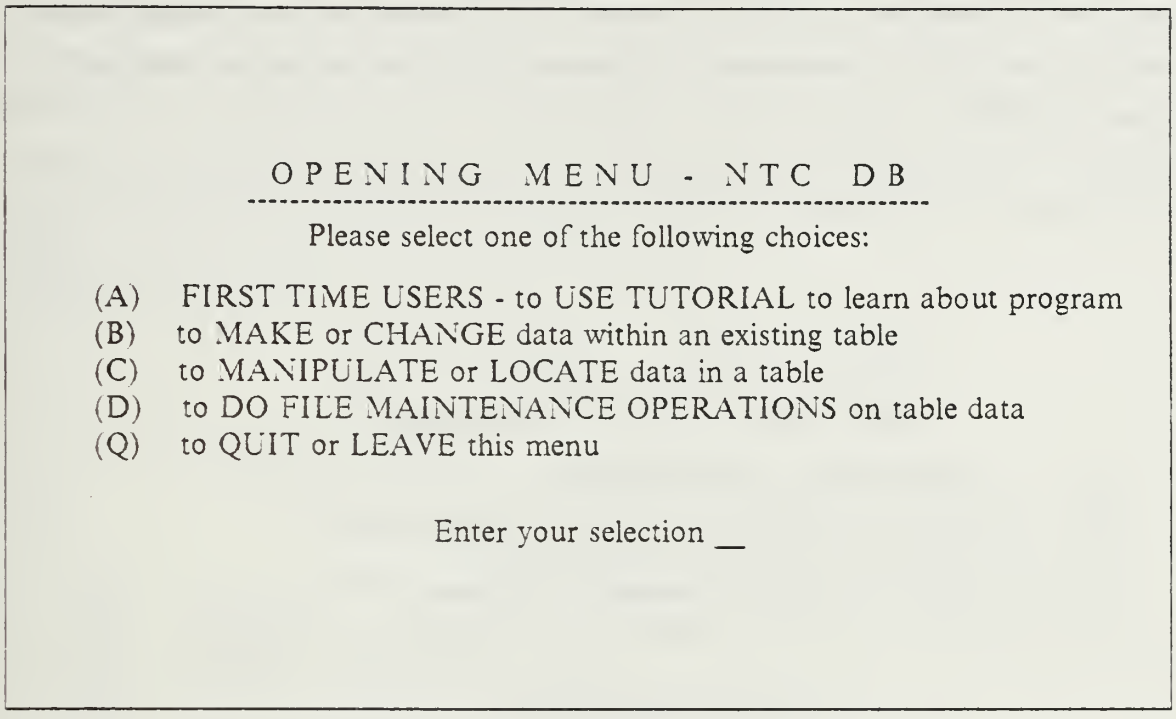


Figure C.6 Initial menu presented to the User.

The three choices that lead the user to the second menu level involve different areas of operations. The second choice from the top menu allows the user to move to a menu that presents choices which allow him to create, change, add, delete, or display data. The third choice from the top menu moves the user to operations which allow him to manipulate or locate data. These include the join, select, find (and membership checks), and analyze data. The fourth choice from the top menu moves the user into the menu which allows him to perform file maintenance operations. These include deleting a table (or file), restoring tables from a backup source, and saving tables to a backup source. Each operation will be elaborated in greater detail within this chapter.

b. **Secondary Level Menus**

The first menu described reflects choice "B" from the initial or "top" menu. As described before, this menu provides operations which allow the user to make or change data (see Figure C.7). The first option from this menu is the operation which can create a table. This operation allows the user to generate a table of his own choice

to fulfill a need to store data. Although the procedure is fairly self-explanatory, a more detailed description will follow in the section dealing with specific operations. The next choice from this menu allows the user to add a tuple of information to an existing table or modify a tuple already in existence. This operation will also be further explained in a follow-on section.

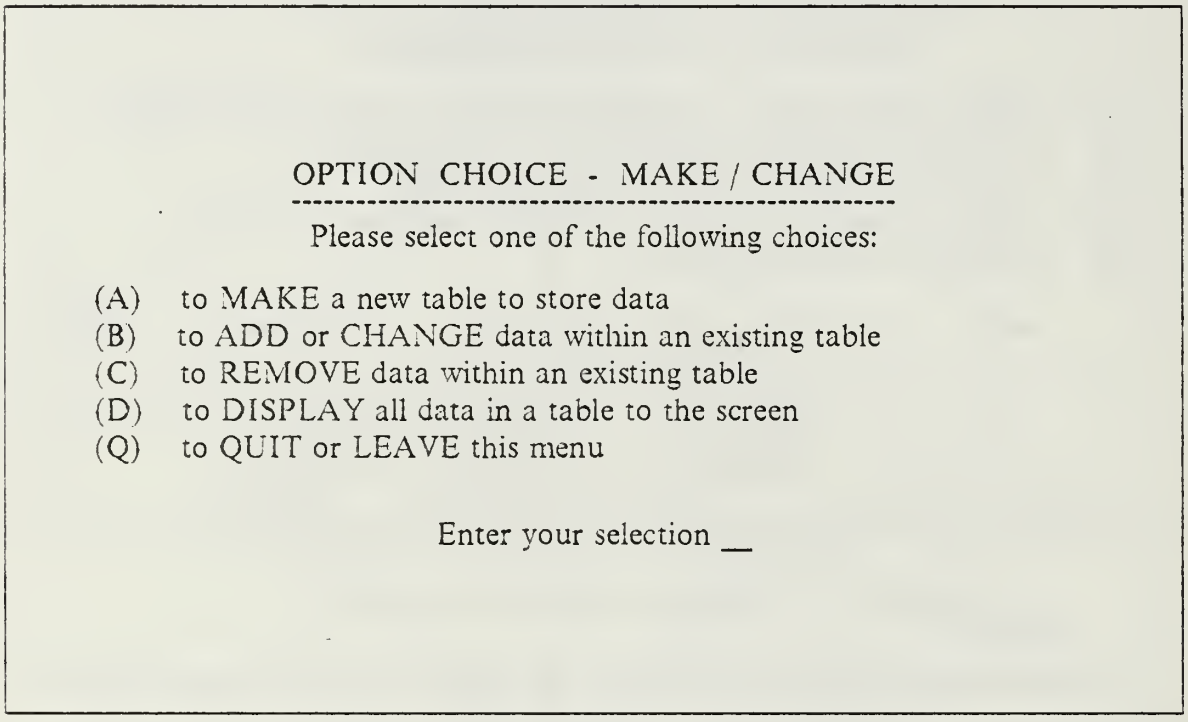


Figure C.7 Option "B" Sublevel Menu.

The next choice from this menu allows the user to delete a tuple of information from one of the database tables. Choice four from this menu presents the user with the opportunity to display, to the monitor screen,a table of information or display the structure of a table. Structure is defined as what attributes are stored within a table and the type of each attribute. The last choice from this menu allows the user to return to the top menu for further choices or to exit the program.

The third choice from the top menu allows the user to access the menu which controls operations that allow the user to manipulate or locate data (see Figure C.8). These operations provide the user with the ability to satisfy query operations. The first choice from this menu provides the capability to join two tables of information together. The only restriction upon this action is that the tables have at least one

common attribute upon which the operation can center. The user is cautioned to invoke this procedure prudently, as the new tables can be very large and cumbersome. The next choice from this menu allows the user to create a new table by selecting attributes from an existing table. This enables the user to tailor tables to meet specific needs. The next operation serves a dual purpose. The find operation allows the user to check for membership within a table and the ability to create a table with a specific value that is common to all tuples. The only user requirement for this operation is that the user know the table where the data is stored.

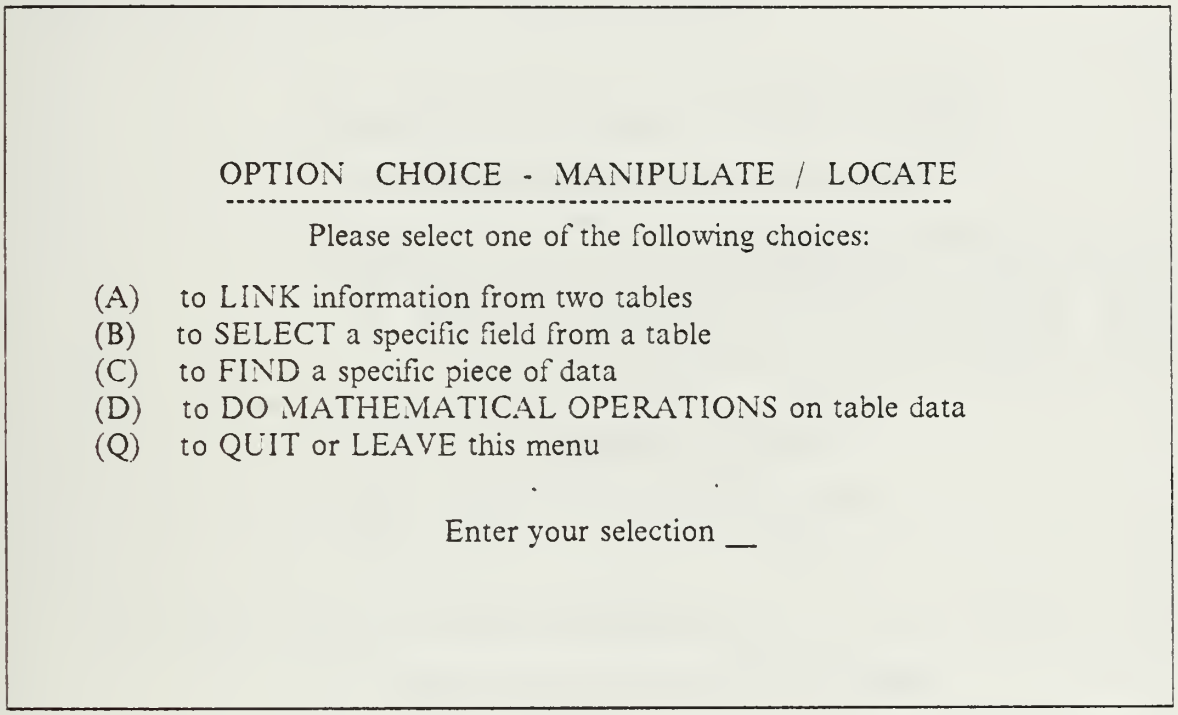


Figure C.8 Option"C" Sublevel Menu.

The fourth choice from this menu provides the user with the capability to do in house analysis of an attribute. This analysis includes the calculation of the mean, range, standard deviation, standard deviation of the mean, and a quartile calculation. These analysis functions are intended to provide a basic analytical capability only and serve as a start for the data analyst. The last choice as in the previous menu, is the "quit" command which returns the user to the top menu for further actions.

The fourth choice from the top menu moves the user to the menu allowing file maintenance operations to be performed. Within this menu the user is provided with four choices. First, he can choose to delete a file or table from the database. Should

the user choose this option he is allowed another chance to exit prior to execution, if a change of mind occurs. The second option allows the user to restore his tables from a backup floppy disk. Like the previous choice, the user is presented an opportunity to exit this option, should a change of mind occur. The third option allows the user to save the data tables he has changed in during the course of the session to his backup floppy. As before, a safety value has been incorporated to prevent hasty use. The last option is the "quit" option which takes the user back to the top menu (see Figure C.9).



Figure C.9 Option "D" Sublevel Menu.

c. **Other Menus used in the Program**

There are several other significant menus which allow the user to progress through the program. These can be broken into two categories. The first category are menus which determine specific operations within an option choice. These allow the user to specify which particular option he wishes to perform and provide a degree of flexibility within that option. An example of such a menu is the menu associated with Mathematical Operations. The option level menu allows the user to choose between summing a column, averaging a column, conducting an analysis of a data column, or counting the number of entries in a table. Each of these menus will be explained in greater detail within the section explaining the option choice.

The next type of menu is the menu which allows the user to choose the category of information that his table is stored under. The intent of this implementation was to provide a logical storage facility under which a user could group his tables. The menu is invoked any time a user is required to choose a table and presents the menu shown in Figure C.10. The categories reflect logical headings which reflect a type of information that the category contains. Any user developed tables which do not "fit" within the confines of one of the category groups can be stored under the catch-all category of user defined tables.

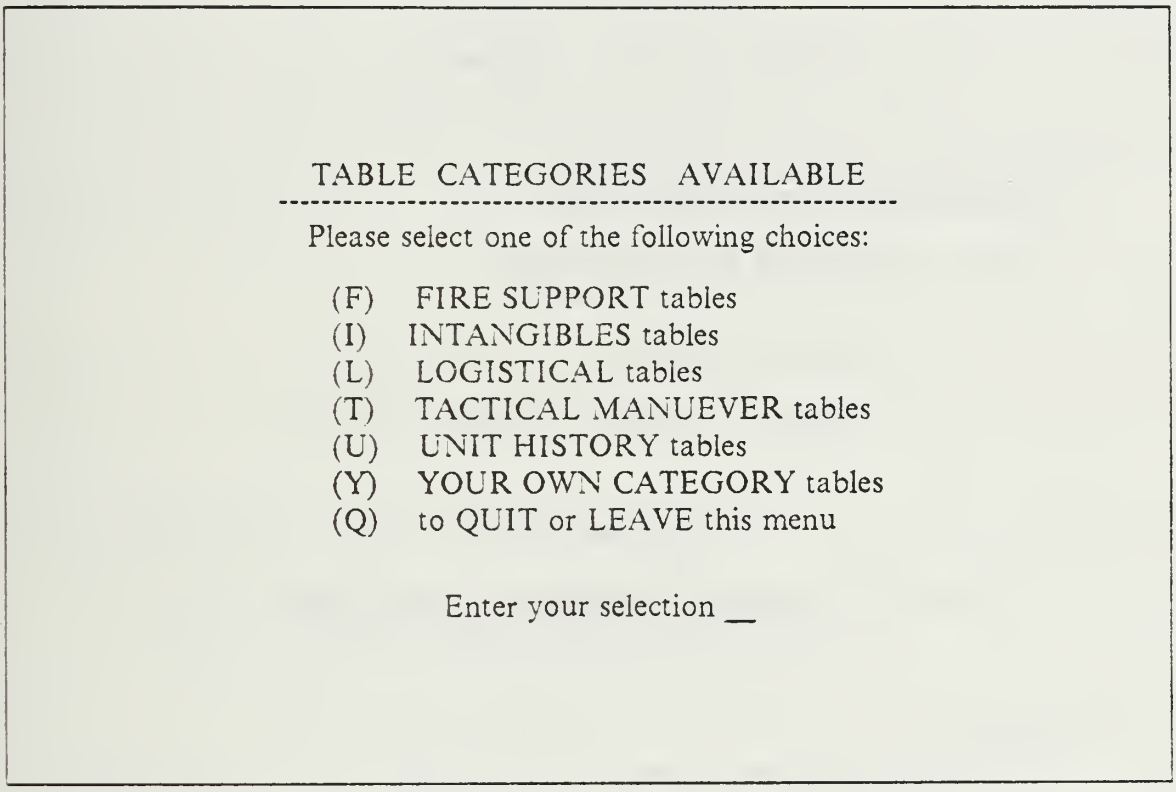


Figure C.10 Category Choice Menu.

d. **Program Operations**

The program presents the user with ten operations from which he can choose. As described in the previous section, the menus are presented with up to five choices within each (one choice being the "quit", to return back to a previous menu). This was implemented in an effort to not overload the user with a large number of choices within one single menu.

The following paragraphs will describe each operation. Within the description of each operation, the manual will highlight specifics which are peculiar to that operation. Menus which are displayed, other than confirmation statements, will be presented and explained. Confirmation statements merely show the reader what choice he has made and give him the option to halt this choice or continue on (see Figure C.11).

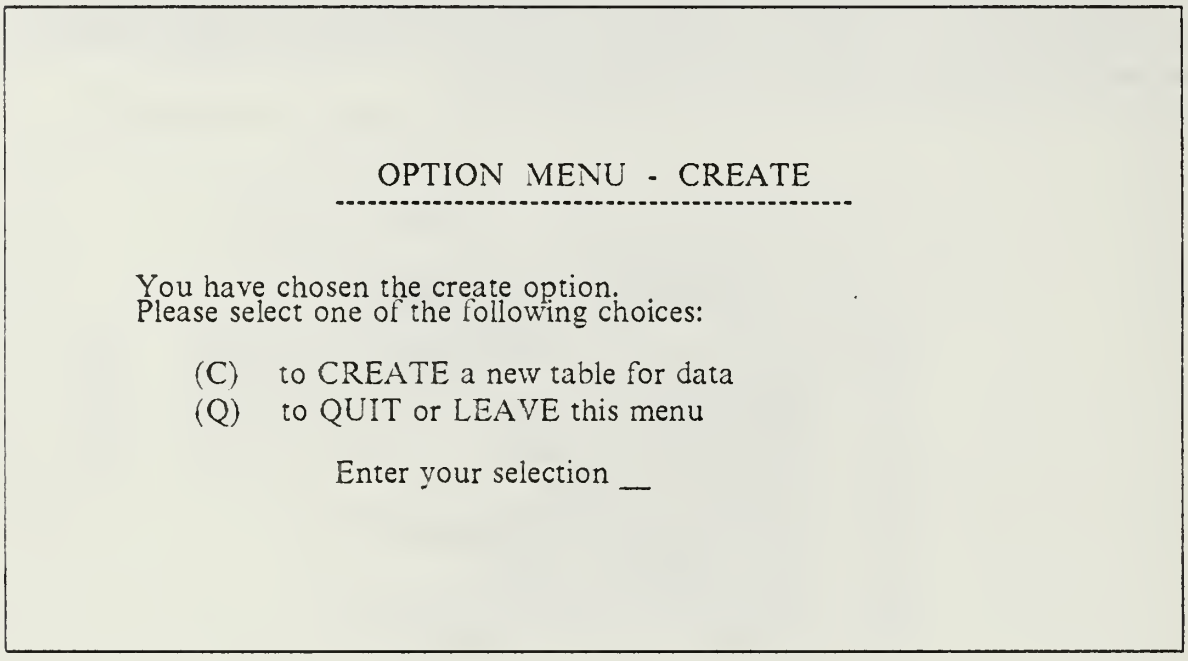


Figure C.11 Example Confirmation Statement Menu.

1. CREATE

The CREATE operation allows the user to design tables of his choosing. The user is allowed to specify the any six letter name for the table. Should that name already be in use, the user is presented with the opportunity to overwrite that table with the new one. Once the user has chosen a name, he is asked to specify the fields of the table and declare a type for each from one the five choices (character, numeric, date, boolean, or memo). After the user has declared a type he is asked to give a numerically length for the entries in that field.

When the user has completed defining the table, he is presented an opportunity to place values within each field. Should he desire to do so, he enters a "y" in the appropriate place and enters data corresponding to the various fields of each

record. Upon completion of this action the user is asked to place the table within a logical storage category. The category should reflect the type of data that the table contains. A "User Defined" category exists to hold tables which do not conform to any other category. When this action has been accomplished, the user is returned to the starting menu.

2. ADD

The ADD option allows the user to add a record of information to the table of his choice. Upon entering this choice from the confirmation menu (see Figure C.12), the user is presented with the category table choices from which he chooses the category of his table. Having chosen the category, he is then presented with the tables in that category from which he can make his table choice. In the event that he has made an incorrect choice or wishes to change, he may enter a 'q' to leave the category and choose another. He may continue in this vain to exit this option choice.

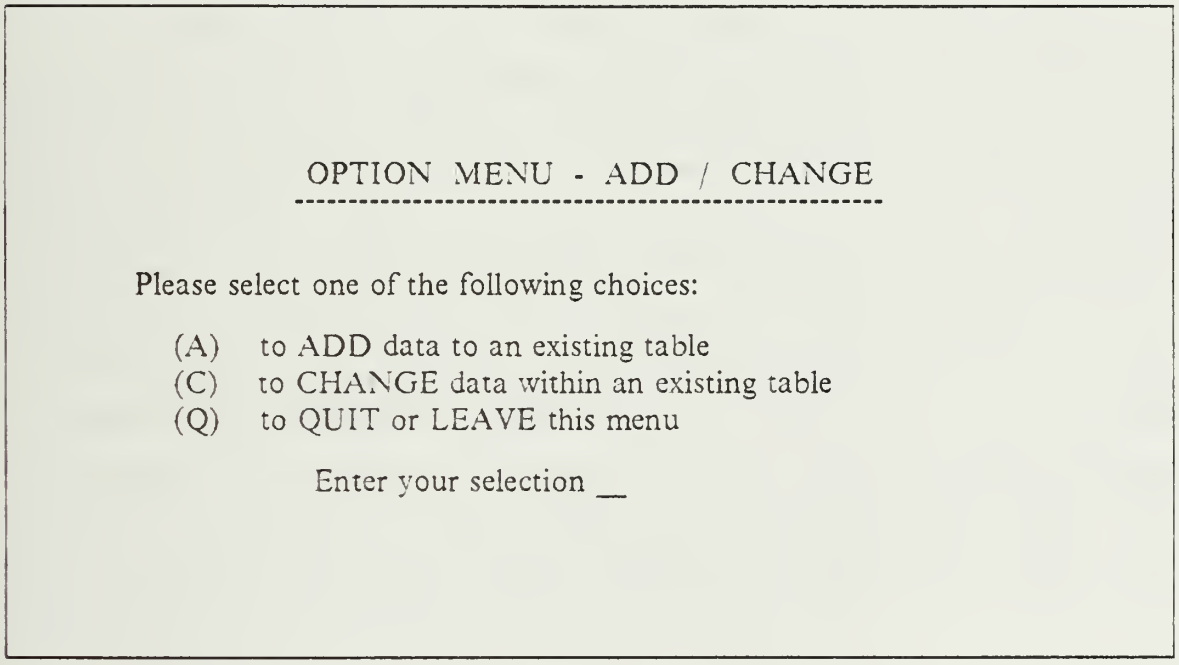


Figure C.12 Add / Change Confirmation Choices.

Assuming that he has picked a table, the user is now presented with the different columns of the record and a highlighted area to the right of each column where he can enter the data. The length of the highlighted area shows how long the entry can be and the user will be moved to each sequentially by pressing the enter key or filling the preceding highlighted area completely. Notice at the top of the page there is a number indicating the corresponding record number that the user is working on.

The user can follow the instructions provided in the menu block to complete his actions or he may press the ENTER key at the beginning of the first highlighted block to exit this addition process. Upon exiting, the user will be placed back at the option menu from which he started.

3. CHANGE

The CHANGE operation allows the user to modify a record which is already in existance. The user has reached this choice by following the same procedure used to reach ADD. At the point where the menu asked for a choice between ADD and CHANGE, the user chose CHANGE (see Figure C.12). The user is then confronted with the category choice menu. After picking the appropriate category that his table resides in, he then enters the table's name in the space provided. At this point should he wish to return to the category choices, he can enter a "Q" in the space reserved for the table name.

Assuming that the user wishes to continue, after picking the table, the records of the table are displayed upon the monitor screen. Using the directional arrow keys, he can move through the records and find the one he wishes to change. Then using the directional keys he can move to the appropriate value and overwrite it in the space provided. Typing the ENTER key stores the value in place. Should the user traverse the entire length of the table, he will be given the option to add data to the table. The user can accomplish this by entering a "Y" after the question. Then as in the add operation he can input the data in the space provided. He exits this mode by typing the ENTER key on a blank field. This will place him at the last value entered where he may perform any changes he wishes or follow the menu information and leave the option (CTRL - END). Should he wish to not save these changes he can abort by pressing the ESC key. In either case, the user will return to the original menu where he can pick another MAKE or CHANGE option.

4. DELETE

The REMOVE option is reached by entering choice "C" within the MAKE or CHANGE menu. This choice allows the user to delete a record from an existing table. Once the user picks this option, he is confronted with another menu which confirms that he wishes to delete a record from a table. Picking the letter "D" continues the action and places the user at the category choice menu. After picking the appropriate category and table name the user is able to begin the operation.

At this point the user is presented with a display of the table of his choice with each record numbered. At the base of the screen, the user is provided a area to enter the respective number of the record that he wishes to delete. Error checking ensures the the input number is within the proper boundaries. Should he wish to not delete any record he is given the last chance to exit by entering a "Q" in the space provided for the record number. Once a number is entered the user can use the SPACE bar to complete the area or press the ENTER key. In each case the record will be deleted and by pressing another key of his choice, he will return to the original menu.

5. DISPLAY

The DISPLAY option is the last choice of the MAKE or CHANGE menu. This option allows the user to list the contents of the table of his choice or display the structure of the table. Once this option has been chosen, the user is confronted with the menu to choose between displaying of the contents or the structure (see Figure C.13). In either case once the user chooses a "D" or an "S" ("Q" returns him to the original menu), he is placed at the category choice menu. After choosing the category, he is then required to enter the name of the table he desires to display.

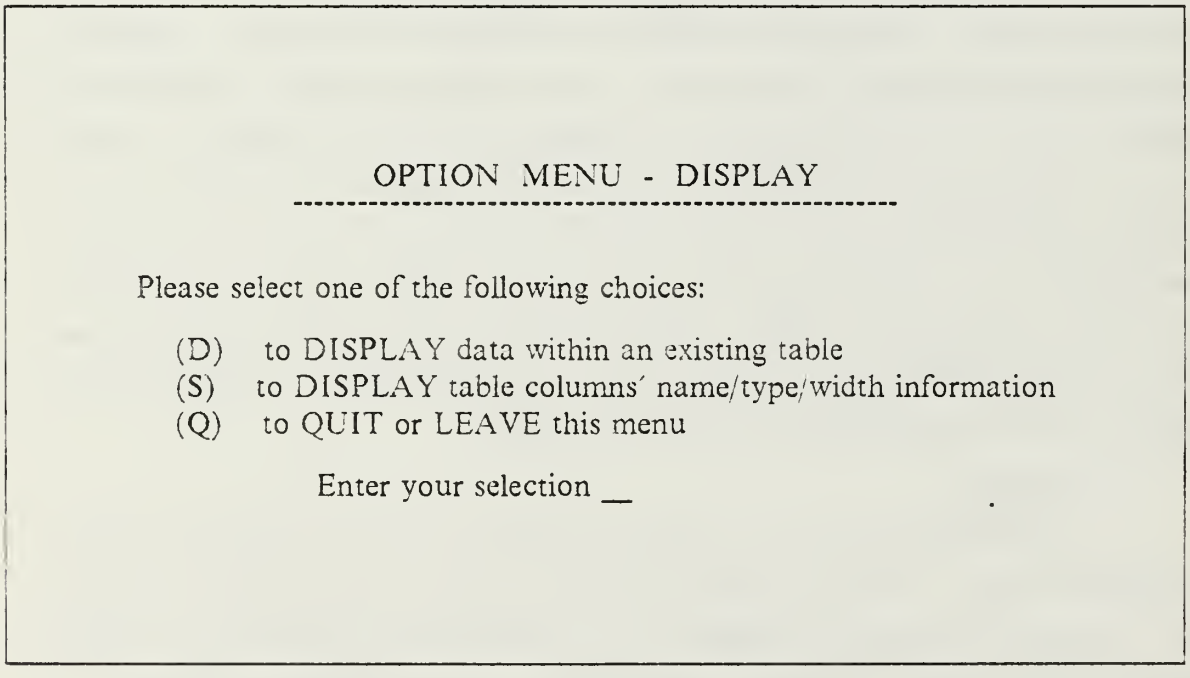


Figure C.13 Display Choice Menu.

If the user had picked a "D" the table contents will be listed on the monitor screen. Should the table be longer than twenty lines, including wrap arounds from wide tables, the user must press a key to continue the display. Once the display is completed, the user is returned to the original menu.

If the user had picked an "S" the different field names of the table will be displayed on the screen. Next to each field name is its respective type, field width, and if it is a numeric, the number of places to the right of the decimal point. By pressing a key of his choice, the user can return to the original menu.

6. JOIN

The Join operation is refered to as the LINK operation to facilitate user understanding. This is because the operation joins or links the data of two tables together based upon a common field. It is important to note that the operation joins on a common field regardless of whether there are common values within that field. In the best case each value of one table has a record in another with that same value, and the new table is the length of the original tables. In the worst case each value from the first table is paired with a value from the second. In this case, the length of the new table is the length of the first TIMES the length of the second. It is for this reason that the user is cautioned when using this operation.

Once the user has decided to use this function, he is asked to pick the first table to be used in the operation. He does this by typing the appropriate category choice and table name. The program will then echo the name of the table which he has picked. He's then given the opportunity to choose the second table in a similar fashion. In either case entering a "Q" in the category choice places him back at the original menu. Once he has chosen both tables the program will ask him to specify the name of his new table. A similar error checking scheme is performed as in the CREATE function and if the table name is not in use, the operation will proceed. Should the name be used the program will place the user back at the start point.

If the table name is not in use, the user is then queried as to the category which he wishes to store his new table under. Once this choice has been made, the operation will continue and the actual linking of the data elements conducted. At the completion of this operation the user will be asked to press any key to place him back at the original menu.

7. SELECT

The SELECT option allows the user to create a table using an existing table. The user does this by specifying the fields of data which he wishes to copy from the old table into his new one. Once he has chosen to continue the operation, he is asked to choose the appropriate table category and name. After picking the table name, the user is provided with a listing of the table to help him determine which fields he wishes to use. At the bottom of the screen, the program will ask him which fields, by name, he wishes to copy. Only a "Y" response copies the field, any other response is interpreted as an "N".

After he has chosen the fields he wishes to copy, the program will ask the user to specify the table name that the data will be stored under. Here again a checking operation similar to the join operation is conducted. If the table name has been used, the user is returned to the start point to try again. If the name is valid, then the user will be asked which category to store the table under. When the operation is completed, the user will be asked to press any key to return to the original menu.

8. FIND

The FIND operation is used in two situations. First, the FIND operation can be used to locate records within a table that contain a specific value. The other reason is to show membership of a value within a table. To utilize the FIND operation, the user must know what value he is looking for and which table is to be scanned. After confirming that he wishes to proceed with this option, the user is asked whether he wishes to retain a table with the information returned from the FIND operation. This allows the user to maintain a permanent record suitable for a query operation. If the user answers "Y" then he must specify a table name under which the data will be stored. The table is automatically conformed to store the data (provided the name passes the error checking scheme). Then the user chooses the new table storage category.

After he has completed the above or should he have answered the question "N", he must pick the appropriate table category and table name. He is then asked to specify the data value which he desires to find an occurrence of in the table. The program will check each field for the value and return those records which contain the value. Should no record be found, a message confirming this will appear on the screen. At the completion of the operation, the user will be asked to press any key to return to

the original menu. If the storage option was chosen, the records displayed will be stored in the new table.

9. MATHEMATICAL OPERATIONS

This option provides the user the capability to accomplish some basic mathematical operations on a column of data. This option is reached through entering a "D" in the MANIPULATE and LOCATE menu. Upon choosing this option, the user is presented with another menu which points out his operation choices (see Figure C.14). The user may SUM the elements of a column, AVERAGE the elements of a column, perform an ANALYSIS of the elements of the column, or he may COUNT the number of records within a table. In each of the first three options the user is allowed to only attempt the operation on a data column of type numeric. The final operation can be used upon any table.

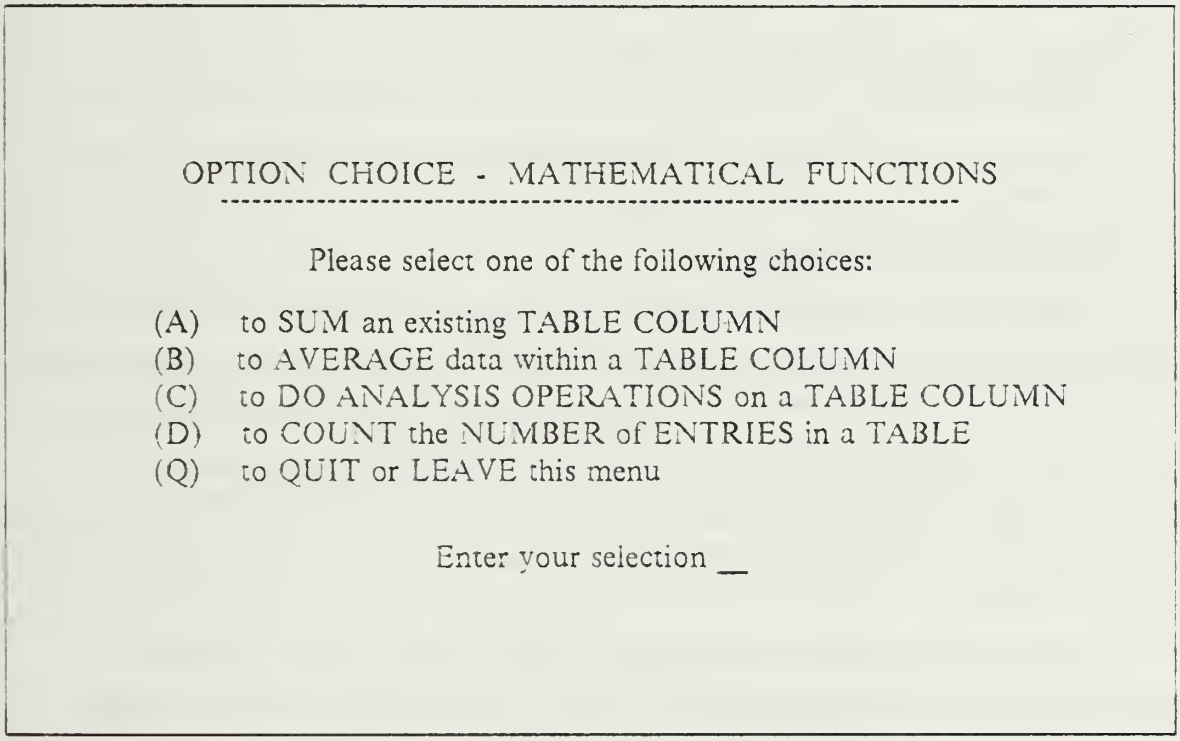


Figure C.14 Mathematical Operations Menu.

The same procedure applies to each of the first three choices, so a general description will be given. In each case, once the user has made a selection of the operation, he is queried for the table category and name. Once the user has identified the table of his choice, a listing of the table is displayed for user convenience. He is

then asked to denote the name of the column upon which he wants to perform the operation. Should he pick a column which is not typed numeric, no action will take place and the entry will be ignored.

When the user has selected a proper data column the operation will act. The SUM operation will return a sum total of all entries in the column. The AVERAGE operation will return a value which represents an average for the data column. The ANALYSIS will return values for the mean, range, standard deviation, standard error of the mean, and a quartile listing. The quartile listing shows the data value which corresponds to a percentile evaluation of the sorted data values. Once these values have been displayed on the screen, the user will be asked to press any key to return to operation choice menu.

The option to calculate the number of records within a table or COUNT begins in a similar manner. The user is queried for the table category and name. Once the table name has been identified, the program returns a value that represents the number of records present in the table. The user is then asked to press any key to return to the operation choice menu. At this menu the user can continue working or enter a "Q" to return back to a previous higher level menu.

10. FILE MAINTENANCE

This option is called in two places. The user can reach this menu by choosing the fourth option from the opening menu, or he can answer "Y" during the QUIT procedure to exit the program. In either situation, the user is presented with a menu of choices which allow him to accomplish the following (see Figure C.14): ERASE a table from memory, RESTORE data from a floppy disk to his tables in main memory, or SAVE a copy of the changes to his tables on a floppy disk and create a backup for the system.

The ERASE operation deletes a table from memory and also removes the table name from the appropriate category. The user is asked to select a table category and name as before. Once the user has selected a table name, the program will ask him again if he wishes to complete this action. Any answer other than a "Y" returns the user to the operation choice menu. A "Y" erases the table and the data is lost from memory. The user is then asked to press any key to return to the operation choice menu.

The RESTORE operation allows the user to recopy information from a backup floppy into main memory. As with any RESTORE operation, the information

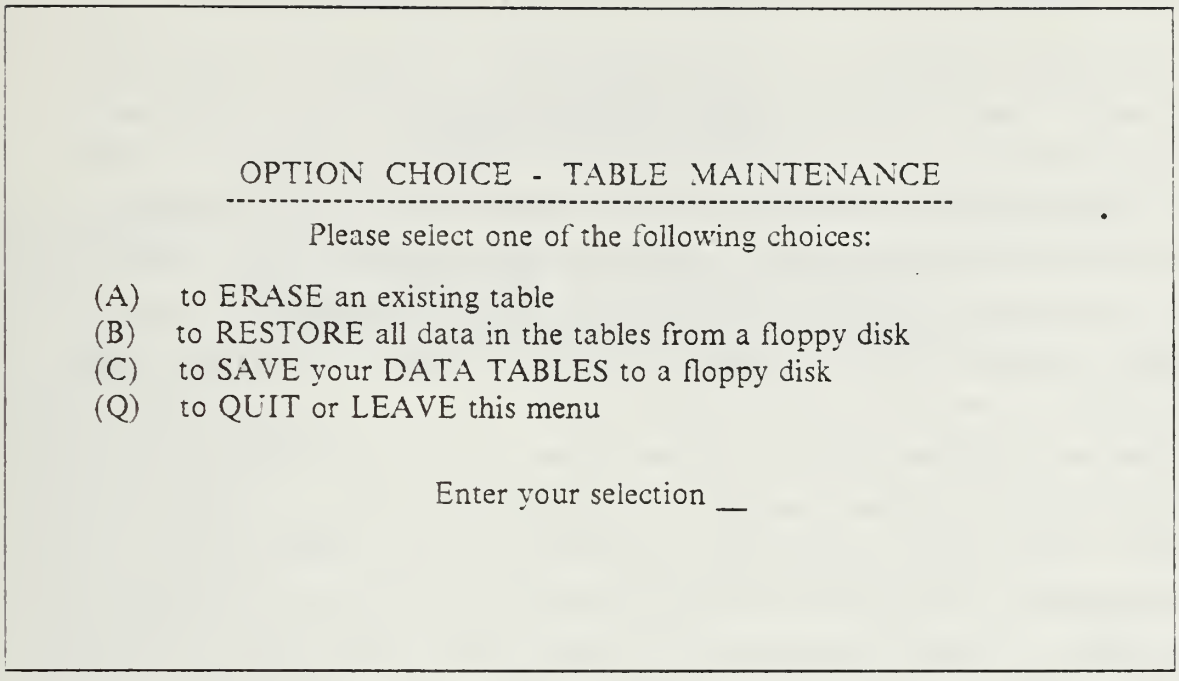


Figure C.14 File Maintenance Operations Menu.

is only as current as the last backup save procedure. Once the user chooses this option he is asked again if he wishes to complete this. The significance of this question is that the RESTORE operation will overwrite any new data within the tables and lose any additions done after the backup save was done. Here again, any answer other than a "Y" returns the user to the operation choice menu.

Should the user wish to continue, he is asked to place his backup floppy into the "B" drive of his system. When the floppy has been placed in the disk drive, the user presses any key to continue. Once the operation is complete, the user presses any key to return to the operation choice menu.

The SAVE operation is just the reversal of the RESTORE operation. In this situation the user is moving a copy of his working tables onto a floppy backup. The user can use any formatted disk to be the backup medium and the "B" drive is used in this procedure also. Once this operation has been completed, the user is asked to press any key to return to the operation choice menu.

4. SUGGESTIONS FOR CONSISTENT PROGRAM PERFORMANCE

The following suggestions are made to help the user maximize the program to its fullest extent. The first suggestion deals with query operations. In all circumstances, the user should attempt to reduce the size (in terms of length and width) of the tables prior to conducting a Join operation. Not only will this speed the operation within the machine, but it will also cut down on extraneous material stored within the table. This can be accomplished by using a Find operation to shorten the number of records (by specifying a particular value if possible). The Select operation can reduce the number of fields within a table to decrease the width of it.

The next suggestion deals with actual queries. To properly implement a query within the program, the user must break the query into operations involving two or less tables. Using Select, Find, and Join operations, the user can whittle away at the query and return a correct answer. The Find operation is perfect for concentrating on specific values. The Select operation keys to specific fields and the Join can be used to link tables together, which in effect links the information upon common bonds.

The final suggestion deals with the table categories. The purpose of the categories is to develop a logical storage mechanism that allows the user to group tables together based on common information. This is provided only as a help to the user, and as an organization tool. The user is encouraged to develop tables of his own and store then appropriately. As more tables become available for use the benefits of these categories will be more fully realized.

A note of caution is provided to the user to remind him of proper operating procedures. It is recommended (and in many areas required) that the user backup his work files daily. This will preclude hours of work attempting to return the system to its current state. The user is also reminded of the storage requirements for his floppy disks (avoid extreme heat and cold, moisture, excessive handling, and dirt). Recent case studies have also shown that these same conditions have effected some microcomputer systems. Ft. Ord, CA reports that a combination of cold morning temperatures and high humidity have caused eractic behavior of their microcomputers.

APPENDIX D
STARTUP PROGRAM

```
rem  ******************************************************
rem  Program: Startup.BAT
rem  Purpose: This program allows the user to properly boot
rem           his own system with the necessary tables
rem           and programs.
rem  ******************************************************

copy B:ntc.bat C:
copy B:ntcmenu.prg C:
copy B:welcome.prg C:
copy B:tutor.txt C:
copy B:browse.* C:
copy B:fs*.dbf C:
copy B:in*.dbf C:
copy B:logistic.dbf C:
copy B:lg*.dbf C:
copy B:ta*.dbf C:
copy B:unithist.dbf C:
copy B:un*.dbf C:
copy B:yourown.dbf C:
copy B:yr*.dbf C:
copy B:procfile.prg C:
copy B:erasor1.bat C:
copy B:savprog.bat C:
copy B:savinto.bat C:
```

APPENDIX E
DATABASE MANAGEMENT PROGRAM CODE

```
rem  ******************************************************
rem  Program: NTC.BAT
rem  Purpose: This program allows the user to properly invoke
rem           the program using DBase III Plus as the outer
rem           program shell.
rem  ******************************************************
DBASE NTCMENU


*************************************************************************
* Program: NTCMENU.prg
* Purpose: The purpose of this program is to provide the
*          user with a mechanism to manipulate bits of data
*          that have been collected at the NTC by giving him
*          the ability to logically store these pieces of
*          data and provide capabilities to move, link, and
*          selectively operate on them through a series of
*          complex functions.
*************************************************************************
*
SET BELL OFF
SET DEVICE TO SCREEN
SET ECHO OFF
SET EXCLUSIVE OFF
SET SAFETY OFF
SET SCOREBOARD OFF
SET STATUS OFF
SET TALK OFF
CLEAR ALL
CLOSE ALL
DO WELCOME
SET PROCEDURE TO PROCFILE
a = .t.
DO WHILE a
    CLEAR
    *
    * Draw menu box
    *
    @ 1,2 to 22,78
    @ 5,20 SAY 'O P E N I N G   M E N U  -  N T C   D B'
    @ 6,18 SAY '-----------------------------------------'
    @ 8,15 SAY 'Please select one of the following choices:'
    @ 10,10 SAY '(A)    FIRST TIME USERS - to USE TUTORIAL'
    @ 10,51 SAY ' to learn about program'
    @ 12,10 SAY '(B)    to MAKE or CHANGE data within an '
    @ 12,50 SAY 'existing table'
    @ 14,10 SAY '(C)    to MANIPULATE or LOCATE data in a table'
    @ 16,10 SAY '(D)    to DO FILE MAINTENANCE OPERATIONS on '
    @ 16,54 SAY 'table data'
    @ 18,10 SAY '(Q)    to QUIT or LEAVE this menu'
    @ 21,20 SAY 'Enter your selection'
    choice = ' '
    DO WHILE .NOT.UPPER(choice)$'ABCDQ'
      @ 21,43 GET choice
      READ
    ENDDO
    CLEAR GETS
    choice = UPPER(choice)
    IF choice='Q'
      SET EXCLUSIVE ON
```

```
      CLEAR
      ans = ' '
      @ 5,10 SAY "If you've made a change within the tables"
      @ 5,51 SAY ' at this time,'
      @ 7,8  SAY 'Would you like to do some file '
      @ 7,39 SAY 'maintenance operations? (Y/N)'
      DO WHILE .NOT. ans$'YNyn'
         @ 8,35 GET ans
         READ
      ENDDO
      IF ans = 'Y' .OR. ans = 'y'
         DO SBMAINT
      ENDIF
      CLEAR
      @ 5,18 SAY "You've left the program at this time."
      @ 8,16 SAY "Thank-you and please come back to use us."
      @ 22,5 SAY "  "
      WAIT
      CLOSE PROCEDURE
      QUIT
   ENDIF
   IF choice='A'
      *
      * To use the tutorial to learn about the program
      *
      CLEAR
      RUN BROWSE.COM TUTOR.TXT
      choice = ' '
   ENDIF
   IF choice='B'
      *
      * To make or change data within a table
      *
      DO SBMENU1
      choice = ' '
   ENDIF
   IF choice='C'
      *
      * To manipulate or locate a specific data element
      *
      DO SBMENU2
      choice = ' '
   ENDIF
   IF choice='D'
      *
      * To file maintenace operations on data tables
      *
      DO SBMAINT
      choice  = ' '
   ENDIF
  a = .t.
ENDDO

*************************************************************************
*  Program : WELCOME.prg
*  Purpose : The purpose of this program is to display a welcome
*            heading for the user to introduce the program.
*************************************************************************
*
* draw welcome message to screen
*
SET STATUS OFF
CLEAR
* draw outside box
*
SET COLOR TO B/W
@ 2,1 to 23,79
@ 6,16 SAY 'N A T I O N A L   T R A I N I N G   C E N T E R'
@ 9,21 SAY 'R E L A T I O N A L   D A T A B A S E '
```

```
@ 12,25 SAY 'D E S I G N   P R O G R A M'
@ 24,10 SAY ' '
WAIT
SET COLOR ON
```

APPENDIX F
PROCEDURE FILE CODE

```
*************************************************************************
* Program: SBMENU1.prg
* Purpose: The purpose of this program is to offer the user,
*          the first second level menu options list.
*************************************************************************
*
PROCEDURE SBMENU1
SET EXCLUSIVE OFF
a = .t.
DO WHILE a
    CLEAR
    *
    * Draw menu box
    *
    @ 1,2 to 22,78
    @ 5,12 SAY 'O P T I O N  C H O I C E -   M A K E  /'
    @ 5,55 SAY ' C H A N G E '
    @ 6,11 SAY '-----------------------------------------'
    @ 6,55 SAY '-------------'
    @ 8,12 SAY 'Please select one of the following choices:'
    @ 10,15 SAY '(A)    to MAKE a new table to store data'
    @ 12,15 SAY '(B)    to ADD or CHANGE data within an existing'
    @ 12,63 SAY 'table'
    @ 14,15 SAY '(C)    to REMOVE data within an existing table'
    @ 16,15 SAY '(D)    to DISPLAY all data in a table to the '
    @ 16,60 SAY 'screen'
    @ 18,15 SAY '(Q)    to QUIT or LEAVE this menu'
    @ 21,20 SAY 'Enter your selection'
    choice = ' '
    DO WHILE .NOT. UPPER(choice)$'ABCDQ'
        @ 21,43 GET choice
        READ
    ENDDO
    CLEAR GETS
    choice = UPPER(choice)
    IF choice='Q'
        SET EXCLUSIVE ON
        CLEAR
        RETURN
    ENDIF
    IF choice='A'
        *
        * To create a new table to store data
        *
        SET SAFETY ON
        SET MENUS ON
        DO SBCREATE
        choice = ' '
    ENDIF
    IF choice='B'
        *
        * To add or change data within a table
        *
        DO SBADD
        choice = ' '
    ENDIF
    IF choice='C'
        *
        * To delete data from an existing table
        *
        DO SBDEL
```

```
        choice = ' '
    ENDIF
    IF choice='D'
        *
        * To display a table to the screen
        *
        DO SBDISPLAY
        choice = ' '
    ENDIF
  a = .t.
ENDDO
*************************************************************************
* Program: SBMENU2.prg
* Purpose: The purpoose of this program is to provide the user
*          with the second level menu options list.
*************************************************************************
*
PROCEDURE SBMENU2
a = .t.
DO WHILE a
    CLEAR
    *
    * Draw menu box
    *
    @ 1,2 to 22,78
    @ 5,7  SAY 'O P T I O N   C H O I C E   -   M A N I P U L A '
    @ 5,55 SAY 'T E  /  L O C A T E'
    @ 6,5  SAY '-------------------------------------------------'
    @ 6,52 SAY '------------------------'
    @ 8,15 SAY 'Please select one of the following choices:'
    @ 10,18 SAY '(A)    to LINK information from two tables'
    @ 12,18 SAY '(B)    to SELECT a specific field from a table'
    @ 14,18 SAY '(C)    to FIND a specific piece of data'
    @ 16,18 SAY '(D)    to DO MATHEMATICAL OPERATIONS on table'
    @ 16,63 SAY ' data'
    @ 18,18 SAY '(Q)    to QUIT or LEAVE this menu'
    @ 21,23 SAY 'Enter your selection'
    choice = ' '
    DO WHILE .NOT. UPPER(choice)$'ABCDQ'
        @ 21,46 GET choice
        READ
    ENDDO
    CLEAR GETS
    choice = UPPER(choice)
    IF choice='Q'
        CLOSE ALL
        CLEAR
        RETURN
    ENDIF
    IF choice='A'
        *
        * To join two tables together
        *
        DO SBJOIN
        choice = ' '
    ENDIF
    IF choice='B'
        *
        * To select a specific field from within a table
        *
        DO SBSELECT
        choice = ' '
    ENDIF
    IF choice='C'
        *
        * To locate a specific piece of information in a table
        *
        DO SBLOCATE
        choice = ' '
    ENDIF
```

```
    IF choice='D'
        *
        * To perform mathematical operations
        *
        DO SBMATH
        choice = ' '
    ENDIF
  a = .t.
ENDDO
*************************************************************************
* Program: SBMAINT.prg
* Purpose: The purpose of this program is to provide the user
*          with file maintenance options to allow his memory
*          space to be more efficiently used and provide a
*          backup system for his information files.
*************************************************************************
*
PROCEDURE SBMAINT
a = .t.
DO WHILE a
    CLEAR
    *
    * Draw menu box
    *
    @ 1,2 to 22,78
    @ 5,9  SAY 'O P T I O N  C H O I C E  -  T A B L E'
    @ 5,51 SAY ' M A I N T E N A N C E'
    @ 6,7 SAY '-------------------------------------'
    @ 6,46 SAY '-----------------------------'
    @ 8,15 SAY 'Please select one of the following choices:'
    @ 10,16 SAY '(A)    to ERASE an existing table'
    @ 12,16 SAY '(B)    to RESTORE all data in the tables from'
    ?? ' a floppy disk '
    @ 14,16 SAY '(C)    to SAVE your DATA TABLES to a floppy disk'
    @ 16,16 SAY '(Q)    to QUIT or LEAVE this menu'
    @ 21,22 SAY 'Enter your selection'
    choice = ' '
    DO WHILE .NOT. UPPER(choice)$'ABCQ'
        @ 21,45 GET choice
        READ
    ENDDO
    CLEAR GETS
    choice = UPPER(choice)
    IF choice='Q'
        CLEAR
        RETURN
    ENDIF
    IF choice='A'
        *
        * To erase an existing table
        *
        SET EXACT ON
        DO TABLOPT
        tablechoice = TRIM(tablechoice)
        tablechoice = UPPER(TRIM(tablechoice))
        IF tablechoice = 'Q' .OR. ASC(tablechoice) = 0
          RETURN
        ENDIF
        response = ' '
        CLEAR
        @ 10, 12 SAY ' CAUTION - are you sure that you want to'
        @ 10, 52 SAY ' DESTROY this table ?'
        @ 12, 27 SAY ' You must enter a Y to continue'
        @ 14, 40 GET response
        READ
        IF response = 'Y' .or. response = 'y'
            CLOSE ALL
            tablename = tablechoice + '.DBF'
            ERASE &tablename
            * erase from appropriate catagory list
```

```
        thrdchoice = UPPER(thrdchoice)
        IF thrdchoice = 'F'
           USE FSUPPRT
           DELETE ALL FOR FSTABLES = tablechoice
           PACK
        ENDIF
        IF thrdchoice = 'I'
           USE INTANG
           DELETE ALL FOR INTABLES = tablechoice
           PACK
        ENDIF
        IF thrdchoice = 'L'
           USE LOGISTIC
           DELETE ALL FOR LGTABLES = tablechoice
           PACK
        ENDIF
        IF thrdchoice = 'T'
           USE TACTIC
           DELETE ALL FOR TATABLES = tablechoice
           PACK
        ENDIF
        IF thrdchoice = 'U'
           USE UNITHIST
           DELETE ALL FOR UHTABLES = tablechoice
           PACK
        ENDIF
        IF thrdchoice = 'Y'
           USE YOUROWN
           DELETE ALL FOR YRTABLES = tablechoice
           PACK
        ENDIF
        WAIT
    ENDIF
    SET EXACT OFF
    choice = ' '
ENDIF
IF choice='B'
    *
    * To backup existing tables from disk
    *
    response = ' '
    CLEAR
    @ 10, 12 SAY ' CAUTION - are you sure that you want to'
    @ 10, 52 SAY ' RESTORE these tables ?'
    @ 12, 27 SAY ' You must enter a Y to continue'
    @ 14, 40 GET response
    READ
    IF response = 'Y' .or. response = 'y'
        CLEAR
        @ 10,23 SAY ' RESTORE OPERATIONS IN PROGRESS'
        @ 12,12 SAY ' You are now recopying all previously '
        @ 12,50 SAY 'stored information'
        @ 14,26 SAY ' into your database files'
        @ 16,20 SAY ' Please place your backup disk in DRIVE B'
        @ 20,20 SAY ' '
        WAIT
        CLOSE DATABASES
        SET SAFETY OFF
        RUN SAVINTO.BAT
        WAIT
    ENDIF
    choice = ' '
ENDIF
IF choice = 'C'
    *
    * To save the existing tables to disk
    response = ' '
    CLEAR
    @ 10, 12 SAY ' CAUTION - are you sure that you want to'
    @ 10, 52 SAY ' SAVE these tables ?'
```

```
        @ 12, 27 SAY ' You must enter a Y to continue'
        @ 14, 40 GET response
        READ
        IF response = 'Y' .or. response = 'y'
           CLEAR
           @ 10,23 SAY ' RECOPY OPERATIONS IN PROGRESS'
           @ 12,12 SAY ' You are now recopying all previously '
           @ 12,50 SAY 'stored information'
           @ 14,26 SAY ' into your disk in the B Drive. '
           @ 16,18 SAY ' Please place a formatted disk into the B Drive'
           @ 21,9 SAY ' '
           WAIT
           RUN SAVPROG.BAT
           WAIT
        ENDIF
        choice = ' '
     ENDIF
     a = .t.
ENDDO
*************************************************************************
* Program: SBCREATE.prg
* Purpose: The purpose of this program is to allow the
*          user to design and save tables of his own
*          choice and imagination.
*************************************************************************
*
PROCEDURE SBCREATE
* Clear all memory locations used to date
*
CLEAR
     *
     @ 5,22 SAY 'O P T I O N   M E N U  -  C R E A T E'
     @ 6,18 SAY '-----------------------------------------'
     @ 8,17 SAY 'You have chosen the create option.'
     @ 9,17 SAY 'Please select one of the following choices:'
     @ 12,20 SAY '(C)    to CREATE a new table for data'
     @ 15,20 SAY '(Q)    to QUIT or LEAVE this menu'
     @ 21,25 SAY 'Enter your selection'
     secnchoice = ' '
     DO WHILE .NOT. UPPER(secnchoice)$'CQ'
        @ 21,48 GET secnchoice
        READ
     ENDDO
     CLEAR GETS
     secnchoice = UPPER(secnchoice)
     IF secnchoice='Q'
        SET EXCLUSIVE ON
        CLEAR
        RETURN
     ENDIF
     IF secnchoice='C'
        *
        * To Create a new table for user data
        CLEAR
        @ 5,10 SAY 'Please enter the name of the table that you'
        @ 5,53 SAY ' wish to create: '
        newtable = '       '
        @ 7,35 GET newtable
        READ
        newtable = UPPER(TRIM(newtable))
        IF newtable = 'Q' .OR. ASC(newtable) = 0
           RETURN
        ENDIF
        * previous table check
        rehash = .F.
        USE FSUPPRT
        prestable = 'FS' + newtable
        LOCATE FOR FSTABLES = prestable
        IF FOUND()
           rehash = .T.
```

```
   locattable = prestable
ENDIF
USE INTANGIB
prestable = 'IN' + newtable
LOCATE FOR INTABLES = prestable
IF FOUND()
   rehash = .T.
   locattable = prestable
ENDIF
USE LOGISTIC
prestable = 'LG' + newtable
LOCATE FOR LGTABLES = prestable
IF FOUND()
   rehash = .T.
   locattable = prestable
ENDIF
USE TACTICS
prestable = 'TA' + newtable
LOCATE FOR TATABLES = prestable
IF FOUND()
   rehash = .T.
   locattable = prestable
ENDIF
USE UNITHIST
prestable = 'UH' + newtable
LOCATE FOR UHTABLES = prestable
IF FOUND()
   rehash = .T.
   locattable = prestable
ENDIF
USE YOUROWN
prestable = 'YR' + newtable
LOCATE FOR YRTABLES = prestable
IF FOUND()
   rehash = .T.
   locattable = prestable
ENDIF
IF .NOT. rehash
   CREATE temp1A.dbf
   check = FILE('temp1A.dbf')
   IF check
       CLEAR
       *
       * Draw one box for table category
       *
       @ 1,2 to 22,78
       @ 5,20 SAY 'C A T E G O R Y   C H O I C E S   '
       @ 6,17 SAY '-------------------------------------'
       @ 8,16 SAY 'Please select one of the following choices'
       @ 9,18 SAY 'to catagorize your table under: '
       @ 11,19 SAY '(F)    FIRE SUPPORT tables '
       @ 12,19 SAY '(I)    INTANGIBLES tables '
       @ 13,19 SAY '(L)    LOGISTICAL tables '
       @ 14,19 SAY '(T)    TACTICAL MANUEVERS tables '
       @ 15,19 SAY '(U)    UNIT HISTORY tables '
       @ 16,19 SAY '(Y)    YOUR OWN CATEGORY tables '
       @ 21,20 SAY 'Enter your selection'
       thrdchoice = ' '
       DO WHILE .NOT. UPPER(thrdchoice)$'FILTUY'
           @ 21, 43 GET thrdchoice
           READ
       ENDDO
       *  place table in category of choice
       thrdchoice = UPPER(thrdchoice)
       IF thrdchoice = 'F'
          USE FSUPPRT
          APPEND BLANK
          newtable = 'FS' + newtable
          REPLACE FSTABLES WITH newtable
       ENDIF
```

```
      IF thrdchoice = 'I'
         USE INTANG
         APPEND BLANK
         newtable = 'IN' + newtable
         REPLACE INTABLES WITH newtable
      ENDIF
      IF thrdchoice = 'L'
         USE LOGISTIC
         APPEND BLANK
         newtable = 'LG' + newtable
         REPLACE LGTABLES WITH newtable
      ENDIF
      IF thrdchoice = 'T'
         USE TACTIC
         APPEND BLANK
         newtable = 'TA' + newtable
         REPLACE TATABLES WITH newtable
      ENDIF
      IF thrdchoice = 'U'
         USE UNITHIST
         APPEND BLANK
         newtable = 'UH' + newtable
         REPLACE UHTABLES WITH newtable
      ENDIF
      IF thrdchoice = 'Y'
         USE YOUROWN
         APPEND BLANK
         newtable = 'YR' + newtable
         REPLACE YRTABLES WITH newtable
      ENDIF
      newtable = newtable + '.DBF'
      COPY FILE temp1A.dbf to &newtable
      newtable = newtable - '.DBF'
      ERASE TEMP1A.DBF
      WAIT
      *
   ENDIF
ENDIF
IF rehash
   CLEAR
   overwrite = ' '
   @ 10,20 SAY ' The table has been previously defined.'
   @ 12,20 SAY ' Do you wish to OVERWRITE it ? (Y/N)'
   @ 12,56 GET overwrite
   READ
   overwrite = UPPER(overwrite)
   IF overwrite = 'Y'
      CREATE temp1A.dbf
      check = FILE('temp1A.dbf')
      IF check
          CLOSE DATABASES
          tablename = locattable + '.DBF'
          ERASE &tablename
          prefix = SUBSTR(locattable,1,2)
          IF prefix = 'FS'
             USE FSUPPRT
             DELETE ALL FOR FSTABLES = locattable
             PACK
          ENDIF
          IF prefix = 'IN'
             USE INTANG
             DELETE ALL FOR INTABLES = locattable
             PACK
          ENDIF
          IF prefix = 'LG'
             USE LOGISTIC
             DELETE ALL FOR LGTABLES = locattable
             PACK
          ENDIF
          IF prefix = 'TA'
```

```
   USE TACTIC
   DELETE ALL FOR TATABLES = locattable
   PACK
ENDIF
IF prefix = 'UH'
   USE UNITHIST
   DELETE ALL FOR UHTABLES = locattable
   PACK
ENDIF
 IF prefix = 'YR'
   USE YOUROWN
   DELETE ALL FOR YRTABLES = locattable
   PACK
ENDIF
CLEAR
*
* Draw menu box
*
@ 1,2 to 22,78
@ 5,20 SAY 'C A T E G O R Y   C H O I C E S   '
@ 6,17 SAY '-------------------------------------'
@ 8,16 SAY 'Please select one of the following choices'
@ 9,18 SAY 'to catagorize your table under: '
@ 11,19 SAY '(F)    FIRE SUPPORT tables '
@ 12,19 SAY '(I)    INTANGIBLES tables '
@ 13,19 SAY '(L)    LOGISTICAL tables '
@ 14,19 SAY '(T)    TACTICAL MANUEVERS tables '
@ 15,19 SAY '(U)    UNIT HISTORY tables '
@ 16,19 SAY '(Y)    YOUR OWN CATEGORY tables '
@ 21,20 SAY 'Enter your selection'
thrdchoice = ' '
DO WHILE .NOT. UPPER(thrdchoice)$'FILTUY'
    @ 21, 43 GET thrdchoice
    READ
ENDDO
*  relocate table in category of choice
thrdchoice = UPPER(thrdchoice)
IF thrdchoice = 'F'
   USE FSUPPRT
   APPEND BLANK
   newtable = 'FS' + newtable
   REPLACE FSTABLES WITH newtable
ENDIF
IF thrdchoice = 'I'
   USE INTANG
   APPEND BLANK
   newtable = 'IN' + newtable
   REPLACE INTABLES WITH newtable
ENDIF
IF thrdchoice = 'L'
   USE LOGISTIC
   APPEND BLANK
   newtable = 'LG' + newtable
   REPLACE LGTABLES WITH newtable
ENDIF
IF thrdchoice = 'T'
   USE TACTIC
   APPEND BLANK
   newtable = 'TA' + newtable
   REPLACE TATABLES WITH newtable
ENDIF
IF thrdchoice = 'U'
   USE UNITHIST
   APPEND BLANK
   newtable = 'UH' + newtable
   REPLACE UHTABLES WITH newtable
ENDIF
IF thrdchoice = 'Y'
   USE YOUROWN
   APPEND BLANK
```

```
          newtable = 'YR' + newtable
          REPLACE YRTABLES WITH newtable
       ENDIF
       newtable = newtable + '.DBF'
       COPY FILE temp1A.dbf to &newtable
       newtable = newtable - '.DBF'
       RUN ERASOR1.BAT
       WAIT
   ENDIF
ENDIF
CLEAR
wish = ' '
@ 10,20 SAY 'The table has been categorized.'
@ 12,20 SAY 'Do you wish to relocate it ? (Y/N)'
@ 12,54 GET wish
READ
wish = UPPER(wish)
IF wish = 'Y'
   prefix = SUBSTR(locattable,1,2)
   IF prefix = 'FS'
      USE FSUPPRT
      DELETE ALL FOR FSTABLES = locattable
      PACK
   ENDIF
   IF prefix = 'IN'
      USE INTANG
      DELETE ALL FOR INTABLES = locattable
      PACK
   ENDIF
   IF prefix = 'LG'
      USE LOGISTIC
      DELETE ALL FOR LGTABLES = locattable
      PACK
   ENDIF
   IF prefix = 'TA'
      USE TACTIC
      DELETE ALL FOR TATABLES = locattable
      PACK
   ENDIF
   IF prefix = 'UH'
      USE UNITHIST
      DELETE ALL FOR UHTABLES = locattable
      PACK
   ENDIF
   IF prefix = 'YR'
      USE YOUROWN
      DELETE ALL FOR YRTABLES = locattable
      PACK
   ENDIF
   CLEAR
   *
   * Draw box for category choice
   *
   @ 1,2 to 22,78
   @ 5,20 SAY 'C A T E G O R Y   C H O I C E S   '
   @ 6,17 SAY '---------------------------------------'
   @ 8,16 SAY 'Please select one of the following choices'
   @ 9,18 SAY 'to catagorize your table under: '
   @ 11,19 SAY '(F)    FIRE SUPPORT tables '
   @ 12,19 SAY '(I)    INTANGIBLES tables '
   @ 13,19 SAY '(L)    LOGISTICAL tables '
   @ 14,19 SAY '(T)    TACTICAL MANUEVERS tables '
   @ 15,19 SAY '(U)    UNIT HISTORY tables '
   @ 16,19 SAY '(Y)    YOUR OWN CATEGORY tables '
   @ 21,20 SAY 'Enter your selection'
   thrdchoice = ' '
   DO WHILE .NOT. UPPER(thrdchoice)$'FILTUY'
      @ 21, 43 GET thrdchoice
      READ
   ENDDO
```

```
            *  relocate with new name into category choice
            thrdchoice = UPPER(thrdchoice)
            IF thrdchoice = 'F'
               USE FSUPPRT
               APPEND BLANK
               newtable = 'FS' + newtable
               REPLACE FSTABLES WITH newtable
               CLOSE DATABASES
               locattable = locattable + '.DBF'
               newtable = newtable + '.DBF'
               RENAME &locattable TO &newtable
            ENDIF
            IF thrdchoice = 'I'
               USE INTANG
               APPEND BLANK
               newtable = 'IN' + newtable
               REPLACE INTABLES WITH newtable
               CLOSE DATABASES
               locattable = locattable + '.DBF'
               newtable = newtable + '.DBF'
               RENAME &locattable TO &newtable
            ENDIF
            IF thrdchoice = 'L'
               USE LOGISTIC
               APPEND BLANK
               newtable = 'LG' + newtable
               REPLACE LGTABLES WITH newtable
               CLOSE DATABASES
               locattable = locattable + '.DBF'
               newtable = newtable + '.DBF'
               RENAME &locattable TO &newtable
            ENDIF
            IF thrdchoice = 'T'
               USE TACTIC
               APPEND BLANK
               newtable = 'TA' + newtable
              REPLACE TATABLES WITH newtable
               CLOSE DATABASES
               locattable = locattable + '.DBF'
               newtable = newtable + '.DBF'
               RENAME &locattable TO &newtable
            ENDIF
            IF thrdchoice = 'U'
               USE UNITHIST
               APPEND BLANK
               newtable = 'UH' + newtable
               REPLACE UHTABLES WITH newtable
               CLOSE DATABASES
               locattable = locattable + '.DBF'
               newtable = newtable + '.DBF'
               RENAME &locattable TO &newtable
            ENDIF
            IF thrdchoice = 'Y'
               USE YOUROWN
               APPEND BLANK
               newtable = 'YR' + newtable
               REPLACE YRTABLES WITH newtable
               CLOSE DATABASES
               locattable = locattable + '.DBF'
               newtable = newtable + '.DBF'
               RENAME &locattable TO &newtable
            ENDIF
         ENDIF
      ENDIF
   ENDIF
   a = .t.
RETURN
*************************************************************************
* Program: SBADD.prg
```

```
* Purpose: Program allows the user to add a record to
*          an existing table or change a record already
*          there.
*************************************************************************
*
PROCEDURE SBADD
* Clear all memory locations used to date
*
CLEAR
    *
    * Draw a menu box
    *
    @1,2 to 22,78
    @ 5,16 SAY 'O P T I O N   M E N U  -  A D D  /  '
    @ 5,52 SAY 'C H A N G E'
    @ 6,13 SAY '---------------------------------------'
    @ 6,45 SAY '---------------------'
    @ 8,12 SAY 'Please select one of the following choices:'
    @ 11,15 SAY '(A)    to ADD data to an existing table'
    @ 13,15 SAY '(C)    to CHANGE data within an existing table'
    @ 15,15 SAY '(Q)    to QUIT or LEAVE this menu'
    @ 21,20 SAY 'Enter your selection'
    secnchoice = ' '
    DO WHILE .NOT. UPPER(secnchoice)$'ACQ'
        @ 21,43 GET secnchoice
        READ
    ENDDO
    CLEAR GETS
    secnchoice = UPPER(secnchoice)
    IF secnchoice='Q'
        SET EXCLUSIVE ON
        CLEAR
        RETURN
    ENDIF
    IF secnchoice='A'
        * Add an entry to a table
        * Must first pick table cataegory
        CLEAR
        @ 5,18 SAY " You've chosen the ADD option"
        @ 16,9 SAY ' '
        WAIT
        DO TABLOPT
        tablechoice = TRIM(tablechoice)
        tablechoice = UPPER(tablechoice)
        IF tablechoice = 'Q' .OR. ASC(tablechoice) = 0
          RETURN
        ENDIF
        USE &tablechoice
        * add entry to table
        APPEND
        CLEAR
        COUNT ALL TO sum
        column = FIELD(1)
        IF TYPE('&column') = 'N'
            blank = VAL(' ')
        ENDIF
        IF TYPE('&column') = 'C'
            blank = ' '
        ENDIF
        IF TYPE('&column') = 'D'
            blank = CTOD(' ')
        ENDIF
        * delete extra blank record if one present
        LOCATE RECORD sum FOR &column = blank
        IF FOUND()
            DELETE RECORD sum
            PACK
        ENDIF
    ENDIF
    IF secnchoice='C'
```

```
        *
        * To change an entry in a table
        * Must first pick table category
        CLEAR
        @ 5,18 SAY " You've chosen the CHANGE option"
        @ 16,9 SAY ' '
        WAIT
        DO TABLOPT
        tablechoice = TRIM(tablechoice)
        tablechoice = UPPER(tablechoice)
        IF tablechoice = 'Q' .OR. ASC(tablechoice) = 0
          RETURN
        ENDIF
        USE &tablechoice
        * cahnge record entries in table
        BROWSE
        PACK
    ENDIF
    a = .t.
RETURN
*************************************************************************
* Program: SBDEL.prg
* Purpose: Program allows the user to remove a record
*          from an existing table.
*************************************************************************
*
PROCEDURE SBDEL
* Clear all memory locations used to date
*
CLEAR
    *
    @ 5,21 SAY 'O P T I O N   M E N U  -  D E L E T E  '
    @ 6,13 SAY '----------------------------------------'
    @ 6,45 SAY '--------------------'
    @ 8,12 SAY 'You have chosen the DELETE option '
    @ 9,12 SAY '  Please choose one of the options below :'
    @ 13,15 SAY '(D)     to DELETE data within an existing table'
    @ 15,15 SAY '(Q)     to QUIT or LEAVE this menu'
    @ 21,20 SAY 'Enter your selection'
    secnchoice = ' '
    DO WHILE .NOT. UPPER(secnchoice)$'DQ'
        @ 21,43 GET secnchoice
        READ
    ENDDO
    CLEAR GETS
    secnchoice = UPPER(secnchoice)
    IF secnchoice='Q'
        SET EXCLUSIVE ON
        CLEAR
        RETURN
    ENDIF
    IF secnchoice='D'
        * Add an entry to a table
        * Must first pick table cataegory
        DO TABLOPT
        tablechoice = TRIM(tablechoice)
        tablechoice = UPPER(tablechoice)
        IF tablechoice = 'Q' .OR. ASC(tablechoice) = 0
           RETURN
        ENDIF
        USE &tablechoice
        IF RECCOUNT() # 0
          CLEAR
          DISPLAY ALL
          ?
          ?
          ?
          * get record number that user wishes to delete
          numrecord = '  '
          @ 21,5 SAY 'Please enter the record number of the line'
```

```
         @ 21,47 SAY ' which you wish to delete :'
         @ 22,25 SAY ' Enter a "q" to delete none.'
         COUNT ALL TO total
         y = .F.
         DO WHILE .NOT. y
            IF VAL(numrecord) <= 0 .OR. VAL(numrecord) > total
                y = .T.
            ENDIF
            IF numrecord = 'q' .OR. numrecord = 'Q'
                y = .T.
            ENDIF
            @ 23, 40 GET numrecord
            READ
         ENDDO
         numrecord = TRIM(numrecord)
         IF numrecord # 'q'.AND. numrecord # 'Q'
            nrecord = VAL(numrecord)
            DELETE RECORD nrecord
            PACK
            WAIT
         ENDIF
      ELSE
         CLEAR
         @ 12, 15 SAY 'You have picked a table with no record'
         ?? ' entries.'
         @ 14, 15 SAY 'Please try this option again with another'
         ?? ' table.'
         @ 20,20 SAY ' '
         WAIT
      ENDIF
   ENDIF
RETURN
*************************************************************************
* Program: SBDISPLAY.prg
* Purpose: Program allows the user to list the record
*          elements of a particular table or display
*          the structure of a table.
*************************************************************************
*
PROCEDURE SBDISPLAY
* Clear all memory locations used to date
*
CLEAR
   *
   * Draw menu box
   *
   @ 1,2 to 22,78
   @ 5,21 SAY 'O P T I O N   M E N U  -  D I S P L A Y'
   @ 6,13 SAY '---------------------------------------'
   @ 6,45 SAY '--------------------'
   @ 9,12 SAY '  Please choose one of the options below :'
   @ 12,15 SAY '(D)    to DISPLAY data within an existing table'
   @ 14,15 SAY '(S)    to DISPLAY table columns name/type/width '
   @ 14,63 SAY 'information'
   @ 16,15 SAY '(Q)    to QUIT or LEAVE this menu'
   @ 21,20 SAY 'Enter your selection'
   secnchoice = ' '
   DO WHILE .NOT. UPPER(secnchoice)$'DSQ'
      @ 21,43 GET secnchoice
      READ
   ENDDO
   CLEAR GETS
   secnchoice = UPPER(secnchoice)
   IF secnchoice='Q'
      SET EXCLUSIVE ON
      CLEAR
      RETURN
   ENDIF
   IF secnchoice='D'
      * To display the table and its contents
```

```
        * Must first pick table cataegory and table
        CLEAR
        @ 5,15 SAY ' You have chosen to display the table contents'
        @ 16,9 SAY ' '
        WAIT
        DO TABLOPT
        tablechoice = TRIM(tablechoice)
        tablechoice = UPPER(tablechoice)
        IF tablechoice = 'Q' .OR. ASC(tablechoice) = 0
          RETURN
        ENDIF
        USE &tablechoice
        CLEAR
        @ 2,10 SAY 'You have chosen the table : '
        @ 2,38 SAY tablechoice
        ?
        DISPLAY OFF ALL
        IF RECCOUNT() = 0
          ? ' You have displayed an empty table.'
          ?
        ENDIF
        WAIT
    ENDIF
    IF secnchoice= 'S'
        * To display the column information or structure
        * Must first pick respective table
        CLEAR
        @ 5,15 SAY ' You have chosen to display the table columns'
        @ 16,9 SAY ' '
        WAIT
        DO TABLOPT
        tablechoice = TRIM(tablechoice)
        tablechoice = UPPER(tablechoice)
        IF tablechoice = 'Q' .OR. ASC(tablechoice) = 0
          RETURN
        ENDIF
        USE &tablechoice
        CLEAR
        @ 2,10 SAY 'You have chosen the table : '
        @ 2,38 SAY tablechoice
        ?
        DISPLAY STRUCTURE
        WAIT
    ENDIF
RETURN


*************************************************************************
* Program: SBJOIN.prg
* Purpose: Program is used to combine two tables worth
*          of information into one using a similar
*          field(s) as the base of the operation.
*************************************************************************
*
PROCEDURE SBJOIN
*
CLEAR
PUBLIC tablechoice
    *
    @ 5,23 SAY 'O P T I O N   M E N U  -  J O I N  '
    @ 6,13 SAY '-----------------------------------'
    @ 6,45 SAY '----------------------'
    @ 8,12 SAY 'You have chosen the JOIN option '
    @ 9,12 SAY '  Please choose one of the options below :'
    @ 13,15 SAY '(J)     to JOIN data within TWO existing tables'
    @ 15,15 SAY '(Q)     to QUIT or LEAVE this menu'
    @ 21,20 SAY 'Enter your selection'
    secnchoice = ' '
    DO WHILE .NOT. UPPER(secnchoice)$'JQ'
```

```
    @ 21,43 GET secnchoice
    READ
ENDDO
CLEAR GETS
secnchoice = UPPER(secnchoice)
IF secnchoice='Q'
    SET EXCLUSIVE ON
    CLEAR
    RETURN
ENDIF
IF secnchoice='J'
    * Select TABLES from an existing tables
    * Must pick FIRST table
    CLOSE ALL
    DO TABLOPT
    choice1 = TRIM(tablechoice)
    choice1 = UPPER(choice1)
    IF choice1 = 'Q' .OR. ASC(choice1) = 0
      RETURN
    ENDIF
    CLEAR
    @ 10,20 SAY 'You have chosen the table : '
    @ 10,48 SAY UPPER(tablechoice)
    @ 21,10 SAY ' '
    WAIT
    * Must pick SECOND table
    DO TABLOPT
    choice2 = TRIM(tablechoice)
    choice2 = UPPER(choice2)
    IF choice2 = 'Q' .OR. ASC(choice2) = 0
      RETURN
    ENDIF
    IF choice1 = choice2
        CLEAR
        @ 12, 20 SAY ' You cannot Join a table with itself.'
        @ 14, 20 SAY ' Please try this option again.'
        @ 20,20 SAY ' '
        WAIT
        RETURN
    ENDIF
    CLEAR
    @ 2,20 SAY 'You have chosen the table : '
    @ 2,48 SAY UPPER(tablechoice)
    @ 7,10 SAY 'Please enter the name of the TABLE which you'
    ?? ' would like'
    @ 9,20 SAY ' this information stored under :'
    newtable = '      '
    @ 11,30 GET newtable
    READ
    newtable = TRIM(newtable)
    newtable = UPPER(newtable)
    IF newtable = 'Q' .OR. ASC(newtable) = 0
      RETURN
    ENDIF
    * previous table check
    rehash = .F.
    USE FSUPPRT
    prestable = 'FS' + newtable
    LOCATE FOR FSTABLES = prestable
    IF FOUND()
      rehash = .T.
    ENDIF
    USE INTANGIB
    prestable = 'IN' + newtable
    LOCATE FOR INTABLES = prestable
    IF FOUND()
      rehash = .T.
    ENDIF
    USE LOGISTIC
    prestable = 'LG' + newtable
```

```
LOCATE FOR LGTABLES = prestable
IF FOUND()
   rehash = .T.
ENDIF
USE TACTICS
prestable = 'TA' + newtable
LOCATE FOR TATABLES = prestable
IF FOUND()
   rehash = .T.
ENDIF
USE UNITHIST
prestable = 'UH' + newtable
LOCATE FOR UHTABLES = prestable
IF FOUND()
   rehash = .T.
ENDIF
USE YOUROWN
prestable = 'YR' + newtable
LOCATE FOR YRTABLES = prestable
IF FOUND()
   rehash = .T.
ENDIF
IF rehash
    CLEAR
    @ 10,20 SAY ' You have chosen a table already in use.'
    @ 12,20 SAY '     Please try this choice again.'
    @ 21,10 SAY ' '
    WAIT
ENDIF
IF  .NOT. rehash
   CLEAR
   * Categorize Operation
   * Draw category menu box
   *
   @ 1,2 to 22,78
   @ 5,20 SAY 'C A T E G O R Y   C H O I C E S   '
   @ 6,17 SAY '--------------------------------------'
   @ 8,16 SAY 'Please select one of the following choices'
   @ 9,18 SAY 'to catagorize your table under: '
   @ 11,19 SAY '(F)    FIRE SUPPORT tables '
   @ 12,19 SAY '(I)    INTANGIBLES tables '
   @ 13,19 SAY '(L)    LOGISTICAL tables '
   @ 14,19 SAY '(T)    TACTICAL MANUEVERS tables '
   @ 15,19 SAY '(U)    UNIT HISTORY tables '
   @ 16,19 SAY '(Y)    YOUR OWN CATEGORY tables '
   @ 21,20 SAY 'Enter your selection'
   thrdchoice = ' '
   DO WHILE .NOT. UPPER(thrdchoice)$'FILTUY'
       @ 21, 43 GET thrdchoice
       READ
   ENDDO
   thrdchoice = UPPER(thrdchoice)
   IF thrdchoice = 'F'
      USE FSUPPRT
      APPEND BLANK
      newtable = 'FS' + newtable
      REPLACE FSTABLES WITH newtable
   ENDIF
   IF thrdchoice = 'I'
      USE INTANG
      APPEND BLANK
      newtable = 'IN' + newtable
      REPLACE INTABLES WITH newtable
   ENDIF
   IF thrdchoice = 'L'
      USE LOGISTIC
      APPEND BLANK
      newtable = 'LG' + newtable
      REPLACE LGTABLES WITH newtable
   ENDIF
```

```
IF thrdchoice = 'T'
   USE TACTIC
   APPEND BLANK
   newtable = 'TA' + newtable
   REPLACE TATABLES WITH newtable
ENDIF
IF thrdchoice = 'U'
   USE UNITHIST
   APPEND BLANK
   newtable = 'UH' + newtable
   REPLACE UHTABLES WITH newtable
ENDIF
IF thrdchoice = 'Y'
   USE YOUROWN
   APPEND BLANK
   newtable = 'YR' + newtable
   REPLACE YRTABLES WITH newtable
ENDIF
* Join Operation
SELECT 2
USE &choice1
num = '1'
number = 1
DO WHILE ASC(FIELD(number)) # 0
    A&num = FIELD(number)
    num = STR(VAL(num) + 1)
    num = LTRIM(num)
    number = number + 1
ENDDO
countA = number - 1
SELECT 1
USE &choice2
numb = '1'
number = 1
DO WHILE ASC(FIELD(number)) # 0
    B&numb = FIELD(number)
    numb = STR(VAL(numb) + 1)
    numb = LTRIM(numb)
    number = number + 1
ENDDO
countB = number - 1
condition = ' '
moverA = 1
moverB = 1
num = '1'
numb = '1'
DO WHILE moverB <= countB
    DO WHILE moverA <= countA
        IF B&numb = A&num
          condition = condition + ',' + B&numb + ' = ' ;
                              + A&num
        ENDIF
        num = STR(VAL(num) + 1)
        num = LTRIM(num)
        moverA = moverA + 1
    ENDDO
    numb = STR(VAL(numb) + 1)
    numb = LTRIM(numb)
    moverB = moverB + 1
ENDDO
l = LEN(condition)
IF l > 1
   l = LEN(condition)
   condition = SUBSTR(condition,3,l - 2)
   JOIN WITH &choice1 TO &newtable FOR &condition
ENDIF
IF l = 1
   CLEAR
   @ 12,20 SAY ' Unable to do JOIN Operation.'
   @ 14,15 SAY ' No compatible columns to join on '
```

```
            ?? 'present.'
            @ 16,20 SAY ' Please try this operation again.'
            @ 20,20 SAY ' '
            WAIT
            IF thrdchoice = 'F'
               USE FSUPPRT
               DELETE ALL FOR FSTABLES = newtable
               PACK
            ENDIF
            IF thrdchoice = 'I'
               USE INTANG
               DELETE ALL FOR INTABLES = newtable
               PACK
            ENDIF
            IF thrdchoice = 'L'
               USE LOGISTIC
               DELETE ALL FOR LGTABLES = newtable
               PACK
            ENDIF
            IF thrdchoice = 'T'
               USE TACTIC
               DELETE ALL FOR TATABLES = newtable
               PACK
            ENDIF
            IF thrdchoice = 'U'
               USE UNITHIST
               DELETE ALL FOR UHTABLES = newtable
               PACK
            ENDIF
            IF thrdchoice = 'Y'
               USE YOUROWN
               DELETE ALL FOR YRTABLES = newtable
               PACK
            ENDIF
            RETURN
         ENDIF
         WAIT
         *
      ENDIF
   ENDIF
RETURN
*************************************************************************
* Program: SBSELECT.prg
* Purpose: The purpose of this program is to allow the
*          user the ability to create a new table using
*          selected columns from an existing table.
*************************************************************************
*
PROCEDURE SBSELECT
*
PUBLIC tablechoice,newtable,fieldlist
CLEAR
   *
   @ 5,21 SAY 'O P T I O N   M E N U  -  S E L E C T '
   @ 6,13 SAY '---------------------------------------'
   @ 6,45 SAY '----------------------'
   @ 8,12 SAY 'You have chosen the SELECT option '
   @ 9,12 SAY '  Please choose one of the options below :'
   @ 13,15 SAY '(S)     to SELECT data within an existing table'
   @ 15,15 SAY '(Q)     to QUIT or LEAVE this menu'
   @ 21,20 SAY 'Enter your selection'
   secnchoice = ' '
   DO WHILE .NOT. UPPER(secnchoice)$'SQ'
      @ 21,43 GET secnchoice
      READ
   ENDDO
   CLEAR GETS
   secnchoice = UPPER(secnchoice)
   IF secnchoice='Q'
      SET EXCLUSIVE ON
```

```
    CLEAR
    RETURN
ENDIF
IF secnchoice='S'
    * Select fields from an existing table
    * Must first pick table cataegory
    DO TABLOPT
    tablechoice = UPPER(tablechoice)
    tablechoice = TRIM(tablechoice)
    IF tablechoice = 'Q' .OR. ASC(tablechoice) = 0
        RETURN
    ENDIF
    USE &tablechoice
    CLEAR
    @ 2,10 SAY 'You have chosen the table : '
    @ 2,38 SAY tablechoice
    ?
    DISPLAY ALL
    ?
    ?
    ?
    * Count Fields
    num = 1
    DO WHILE ASC(FIELD(num)) # 0
      num = num + 1
    ENDDO
    num = num - 1
    colindex = 1
    response = '0,'
    * user response for columns to copy
    DO WHILE  colindex <= num
      @ 22 ,9 SAY '                                                '
      @ 22 ,50 SAY '                                               '
      @ 22,10 SAY 'Do you wish to copy column '
      ?? FIELD(colindex)
      ?? ' (Y/N)'
      answer = ' '
      @ 23, 30 GET answer
      READ
      IF answer = 'y' .or. answer = 'Y'
        response = response + LTRIM(STR(colindex)) + ','
      ENDIF
      IF answer = 'q' .or. answer = 'Q'
        colindex = num + 1
      ENDIF
      colindex = colindex + 1
    ENDDO
    CLEAR
    IF LEN(response) = 2
      @ 12,10 SAY ' You have decided to NOT SELECT any column'
    ENDIF
    newtable = '      '
    fieldlist = ''
    IF LEN(response) > 2
      l = LEN(response)
      response = SUBSTR( response,3,l - 2)
      @ 12, 10 SAY ' PLease enter the name of the new table'
      @ 14 , 27 GET newtable
      READ
      newtable = TRIM(newtable)
      newtable = UPPER(newtable)
      IF newtable = 'Q' .OR. ASC(newtable) = 0
        RETURN
      ENDIF
      * previous table check
      rehash = .F.
      USE FSUPPRT
      prestable = 'FS' + newtable
      LOCATE FOR FSTABLES = prestable
      IF FOUND()
```

```
   rehash = .T.
ENDIF
USE INTANGIB
prestable = 'IN' + newtable
LOCATE FOR INTABLES = prestable
IF FOUND()
   rehash = .T.
ENDIF
USE LOGISTIC
prestable = 'LG' + newtable
LOCATE FOR LGTABLES = prestable
IF FOUND()
   rehash = .T.
ENDIF
USE TACTICS
prestable = 'TA' + newtable
LOCATE FOR TATABLES = prestable
IF FOUND()
   rehash = .T.
ENDIF
USE UNITHIST
prestable = 'UH' + newtable
LOCATE FOR UHTABLES = prestable
IF FOUND()
   rehash = .T.
ENDIF
USE YOUROWN
prestable = 'YR' + newtable
LOCATE FOR YRTABLES = prestable
IF FOUND()
   rehash = .T.
ENDIF
IF rehash
    CLEAR
    @ 10,20 SAY 'You have chosen a table already in use.'
    @ 12,20 SAY '    Please try this choice again.'
    @ 21,10 SAY ' ' .
    WAIT
ENDIF
IF .NOT. rehash
   CLEAR
   * Do SELECT Operation
   *
   CLEAR
   @ 7, 12 SAY ' Please identify the catagory for your'
   @ 7, 50 SAY ' new table :'
   @ 11,19 SAY '(F)    FIRE SUPPORT tables '
   @ 12,19 SAY '(I)    INTANGIBLES tables '
   @ 13,19 SAY '(L)    LOGISTICAL tables '
   @ 14,19 SAY '(T)    TACTICAL MANUEVERS tables '
   @ 15,19 SAY '(U)    UNIT HISTORY tables '
   @ 16,19 SAY '(Y)    YOUR OWN CATEGORY tables '
   @ 21,20 SAY 'Enter your selection'
   thrdchoice = ' '
   DO WHILE .NOT. UPPER(thrdchoice)$'FILTUY'
       @ 21, 43 GET thrdchoice
       READ
   ENDDO
   * insert table into category choice
   thrdchoice = UPPER(thrdchoice)
   IF thrdchoice = 'F'
      USE FSUPPRT
      APPEND BLANK
      newtable = 'FS' + newtable
      REPLACE FSTABLES WITH newtable
   ENDIF
   IF thrdchoice = 'I'
      USE INTANG
      APPEND BLANK
      newtable = 'IN' + newtable
```

```
            REPLACE INTABLES WITH newtable
          ENDIF
          IF thrdchoice = 'L'
            USE LOGISTIC
            APPEND BLANK
            newtable = 'LG' + newtable
            REPLACE LGTABLES WITH newtable
          ENDIF
          IF thrdchoice = 'T'
            USE TACTIC
            APPEND BLANK
            newtable = 'TA' + newtable
            REPLACE TATABLES WITH newtable
          ENDIF
          IF thrdchoice = 'U'
            USE UNITHIST
            APPEND BLANK
            newtable = 'UH' + newtable
            REPLACE UHTABLES WITH newtable
          ENDIF
          IF thrdchoice = 'Y'
            USE YOUROWN
            APPEND BLANK
            newtable = 'YR' + newtable
            REPLACE YRTABLES WITH newtable
          ENDIF
          CLOSE DATABASES
          WAIT
          USE &tablechoice
          DO WHILE ASC(response) # 0
              newfield = VAL(SUBSTR(response,1,1))
              FieldId = FIELD(newfield)
              fieldlist = fieldlist + FieldId + ','
              l = LEN(response)
              response = SUBSTR( response,3,l - 2)
          ENDDO
          l = LEN(fieldlist)
          fieldlist = SUBSTR(fieldlist,1,l - 1)
          newtable = UPPER(newtable)
          COPY TO &newtable FIELD &fieldlist
          * display result to screen
          USE &newtable
          ? '       For the new table '
          ?? newtable
          ?? ' the listing is :'
          ?
          ?
          DISPLAY ALL
          WAIT
          *
        ENDIF
      ENDIF
    ENDIF
RETURN
*************************************************************************
* Program: SBLOCATE.prg
* Purpose: Program is used to locate a specific data
*          element within a database table.
*************************************************************************
*
PROCEDURE SBLOCATE
*
SET EXACT OFF
CLEAR
    *
    @ 5,21 SAY 'O P T I O N   M E N U  -  L O C A T E '
    @ 6,13 SAY '---------------------------------------'
    @ 6,45 SAY '--------------------'
    @ 8,12 SAY 'You have chosen the LOCATE option '
    @ 9,12 SAY '  Please choose one of the options below :'
```

```
@ 13,15 SAY '(L)     to LOCATE data within an existing table'
@ 15,15 SAY '(Q)     to QUIT or LEAVE this menu'
@ 21,20 SAY 'Enter your selection'
secnchoice = ' '
DO WHILE .NOT. UPPER(secnchoice)$'LQ'
    @ 21,43 GET secnchoice
    READ
ENDDO
CLEAR GETS
secnchoice = UPPER(secnchoice)
IF secnchoice='Q'
    SET EXCLUSIVE ON
    CLEAR
    RETURN
ENDIF
IF secnchoice='L'
    *
    * Determine if user wants a permanant copy
    CLEAR
    copychoice = ' '
    DO WHILE .NOT. UPPER(copychoice)$'YNQ'
        @ 10,15 SAY 'Do you wish to retain a copy of this'
        ?? ' output?'
        @ 12,15 SAY 'Please enter your decision (Y/N): '
        @ 12,49 GET copychoice
        READ
    ENDDO
    rehash = .F.
    IF UPPER(copychoice) = 'Y'
        newtable = '       '
        @ 15,15 SAY 'Please enter the name of the table '
        ?? 'you wish to use'
        @ 16,35 GET newtable
        READ
        newtable = TRIM(newtable)
        newtable = UPPER(newtable)
        IF newtable = 'Q' .OR. ASC(newtable) = 0
          RETURN
        ENDIF
        * previous table check
        USE FSUPPRT
        prestable = 'FS' + newtable
        LOCATE FOR FSTABLES = prestable
        IF FOUND()
          rehash = .T.
        ENDIF
        USE INTANGIB
        prestable = 'IN' + newtable
        LOCATE FOR INTABLES = prestable
        IF FOUND()
          rehash = .T.
        ENDIF
        USE LOGISTIC
        prestable = 'LG' + newtable
        LOCATE FOR LGTABLES = prestable
        IF FOUND()
          rehash = .T.
        ENDIF
        USE TACTICS
        prestable = 'TA' + newtable
        LOCATE FOR TATABLES = prestable
        IF FOUND()
          rehash = .T.
        ENDIF
        USE UNITHIST
        prestable = 'UH' + newtable
        LOCATE FOR UHTABLES = prestable
        IF FOUND()
          rehash = .T.
        ENDIF
```

```
USE YOUROWN
prestable = 'YR' + newtable
LOCATE FOR YRTABLES = prestable
IF FOUND()
  rehash = .T.
ENDIF
IF rehash
    CLEAR
    @ 10,20 SAY 'You have chosen a table already in use.'
    @ 12,20 SAY '    Please try this choice again.'
    @ 21,10 SAY ' '
    WAIT
    RETURN
ENDIF
IF .NOT. rehash
   CLEAR
   * Do LOCATE Operation
   *
   CLEAR
   @ 7, 12 SAY ' Please identify the catagory for your'
   @ 7, 50 SAY ' new table :'
   @ 11,19 SAY '(F)    FIRE SUPPORT tables '
   @ 12,19 SAY '(I)    INTANGIBLES tables '
   @ 13,19 SAY '(L)    LOGISTICAL tables '
   @ 14,19 SAY '(T)    TACTICAL MANUEVERS tables '
   @ 15,19 SAY '(U)    UNIT HISTORY tables '
   @ 16,19 SAY '(Y)    YOUR OWN CATEGORY tables '
   @ 21,20 SAY 'Enter your selection'
   thdchoice = ' '
   DO WHILE .NOT. UPPER(thdchoice)$'FILTUY'
       @ 21, 43 GET thdchoice
       READ
   ENDDO
   thdchoice = UPPER(thdchoice)
   IF thdchoice = 'F'
      USE FSUPPRT
      APPEND BLANK
      newtable = 'FS' + newtable
      REPLACE FSTABLES WITH newtable
   ENDIF
   IF thdchoice = 'I'
      USE INTANG
      APPEND BLANK
      newtable = 'IN' + newtable
      REPLACE INTABLES WITH newtable
   ENDIF
   IF thdchoice = 'L'
      USE LOGISTIC
      APPEND BLANK
      newtable = 'LG' + newtable
      REPLACE LGTABLES WITH newtable
   ENDIF
   IF thdchoice = 'T'
      USE TACTIC
      APPEND BLANK
      newtable = 'TA' + newtable
      REPLACE TATABLES WITH newtable
   ENDIF
   IF thdchoice = 'U'
      USE UNITHIST
      APPEND BLANK
      newtable = 'UH' + newtable
      REPLACE UHTABLES WITH newtable
   ENDIF
   IF thdchoice = 'Y'
      USE YOUROWN
      APPEND BLANK
      newtable = 'YR' + newtable
      REPLACE YRTABLES WITH newtable
   ENDIF
```

```
      CLOSE DATABASES
      WAIT
      *
    ENDIF
ENDIF
* To locate data from an existing table
* Must first pick table
IF .NOT. rehash
   DO TABLOPT
   tablechoice = UPPER(tablechoice)
   tablechoice = TRIM(tablechoice)
   IF tablechoice = 'Q' .OR. ASC(tablechoice) = 0
       RETURN
   ENDIF
   USE &tablechoice
   IF UPPER(copychoice) = 'Y'
       COPY STRUCTURE TO &newtable
   ENDIF
   CLEAR
   @ 2,10 SAY 'You have chosen the table : '
   @ 2,38 SAY tablechoice
   data = '                    '
   @ 5,10 SAY  'Please enter the data which you would like found'
   @ 6,25 GET data
   READ
   * search for value, compare against all values within table
   numfields = 1
   DO WHILE ASC(FIELD(numfields)) # 0
       numfields = numfields + 1
   ENDDO
   numfields = numfields - 1
   c = 0
   checkval = 0
   cnter1 = 1
   numeric = VAL(data)
   COUNT ALL TO numrecords
   DO WHILE cnter1 <= numfields
      cnter2 = 1
      colchoice = FIELD(cnter1)
      DO WHILE cnter2 <= numrecords
          DO CASE
             CASE TYPE('&colchoice') = 'C'
                cdata = TRIM(data)
                LOCATE RECORD cnter2 FOR &colchoice = cdata
                IF FOUND()
                   LIST RECORD cnter2
                   IF UPPER(copychoice) = 'Y'.AND. checkval < cnter1
                      USE &newtable
                      APPEND FROM &tablechoice FOR &colchoice = cdata
                      checkval = cnter1
                      USE &tablechoice
                   ENDIF
                   c = c + 1
                ENDIF
             CASE TYPE('&colchoice') = 'D'
                IF SUBSTR(DATA,3,1) = '/'
                   STORE CTOD(data) TO ddata
                   LOCATE RECORD cnter2 FOR &colchoice = ddata
                   IF FOUND()
                      LIST RECORD cnter2
                      IF UPPER(copychoice) = 'Y'.AND. checkval < cnter1
                         USE &newtable
                         APPEND FROM &tablechoice FOR &colchoice = ddata
                         checkval = cnter1
                         USE &tablechoice
                      ENDIF
                      c = c + 1
                   ENDIF
                ENDIF
             CASE TYPE('&colchoice') = 'N'
```

```
                    LOCATE RECORD cnter2 FOR &colchoice = numeric
                    IF FOUND()
                      LIST RECORD cnter2
                      IF UPPER(copychoice) = 'Y'.AND. checkval < cnter1
                        USE &newtable
                        APPEND FROM &tablechoice FOR &colchoice = numeric
                        checkval = cnter1
                        USE &tablechoice
                      ENDIF
                      c = c + 1
                    ENDIF
                  CASE TYPE('&colchoice') = 'L'
                    IF LEN(data) = 1
                      IF data = 'T'
                        boolean = .T.
                      ENDIF
                      IF data = 'F'
                        boolean = .F.
                      ENDIF
                      LOCATE RECORD cnter2 FOR &colchoice = boolean
                      IF FOUND()
                        LIST RECORD cnter2
                        IF UPPER(copychoice) = 'Y'.AND. checkval < cnter1
                          USE &newtable
                          APPEND FROM &tablechoice FOR &colchoice = boolean
                          checkval = cnter1
                          USE &tablechoice
                        ENDIF
                        c = c + 1
                      ENDIF
                    ENDIF
                ENDCASE
                cnter2 = cnter2 + 1
            ENDDO
            cnter1 = cnter1 + 1
          ENDDO
          IF c = 0
            ?  ' No records are listed because the data was not'
            ?? ' present.'
          ENDIF
          WAIT
        ENDIF
      ENDIF
RETURN
*******************************************************************************
* Program: SBMATH.prg
* Purpose: The purpose of this program is to provide the user
*          with mathematical options from which he may choose
*          to perform on specific fields in a file.
*******************************************************************************
*
PROCEDURE SBMATH
a = .t.
DO WHILE a
    CLEAR
    *
    * Draw menu box
    *
    @ 1,2 to 22,78
    @ 3,4  SAY 'O P T I O N   C H O I C E  -  M A T H E'
    @ 3,43 SAY ' M A T I C A L   F U N C T I O N S'
    @ 6,4 SAY '-------------------------------------'
    @ 6,42 SAY '-------------------------------------'
    @ 8,15 SAY 'Please select one of the following choices:'
    @ 10,16 SAY '(A)    to SUM an existing TABLE COLUMN'
    @ 12,16 SAY '(B)    to AVERAGE data within a TABLE COLUMN '
    @ 14,16 SAY '(C)    to DO ANALYSIS OPERATIONS on a TABLE '
    ?? 'COLUMN'
    @ 16,16 SAY '(D)    to COUNT the NUMBER of ENTRIES in a TABLE'
    @ 18,16 SAY '(Q)    to QUIT or LEAVE this menu'
```

```
@ 21,22 SAY 'Enter your selection'
choice = ' '
DO WHILE .NOT. UPPER(choice)$'ABCDQ'
    @ 21,45 GET choice
    READ
ENDDO
CLEAR GETS
choice = UPPER(choice)
IF choice='Q'
    CLEAR
    RETURN
ENDIF
IF choice='A'
    *
    * To sum a column from a table
    *
    DO TABLOPT
    tablechoice = UPPER(tablechoice)
    tablechoice = TRIM(tablechoice)
    IF tablechoice = 'Q' .OR. ASC(tablechoice) = 0
        RETURN
    ENDIF
    USE &tablechoice
    nonumeric = .T.
    i = 1
    DO WHILE ASC(FIELD(i)) # 0
        column = FIELD(i)
        IF TYPE('&column') = 'N'
            nonumeric = .F.
        ENDIF
        i = i + 1
    ENDDO
    IF nonumeric
        ?
        ? ' The table you have chosen has no numeric columns'
        ?? ' to sum.'
        ?
        ?
        WAIT
        RETURN
    ENDIF
    CLEAR
    DISPLAY OFF ALL
    colchoice = '          '
    ?
    ?
    ?
    @ 22,10 SAY ' Choose the column you wish summed:'
    x = .F.
    num = 1
    * if column "numeric" do operation
    DO WHILE .NOT. x
        @ 22,46 GET colchoice
        READ
        colchoice = TRIM(UPPER(colchoice))
        y = .F.
        DO WHILE .NOT. y
        IF FIELD(num) = colchoice .and. asc(colchoice) > 0
           IF TYPE('&colchoice') =  'N'
              x = .T.
              y = .T.
              SUM ALL &colchoice TO sumtotal
              ? '     The sum of that column is'
              ?? sumtotal
              ?
              ?
              WAIT
           ENDIF
        ENDIF
        num = num + 1
```

```
        IF asc(field(num)) = 0
           y = .T.
           num = 1
           colchoice = '            '
        ENDIF
        ENDDO
    ENDDO
    choice = ' '
ENDIF
IF choice='B'
    *
    * To average the data values of a table's column
    *
    DO TABLOPT
    tablechoice = UPPER(tablechoice)
    tablechoice = TRIM(tablechoice)
    IF tablechoice = 'Q' .OR. ASC(tablechoice) = 0
        RETURN
    ENDIF
    USE &tablechoice
    nonumeric = .T.
    i = 1
    DO WHILE ASC(FIELD(i)) # 0
        column = FIELD(i)
        IF TYPE('&column') = 'N'
           nonumeric = .F.
        ENDIF
        i = i + 1
    ENDDO
    IF nonumeric
        ?
        ? ' The table you have chosen has no numeric columns'
        ?? ' to average.'
        ?
        ?
        WAIT
        RETURN
    ENDIF
    CLEAR
    DISPLAY OFF ALL
    ?
    ?
    ?
    colchoice = '            '
    @ 22, 10 SAY ' Choose the column you wish averaged:'
    x = .F.
    * if column "numeric" do operation
    num = 1
    DO WHILE .NOT. x
        @ 22,46 GET colchoice
        READ
        colchoice = TRIM(UPPER(colchoice))
        y = .F.
        DO WHILE .NOT. y
        IF FIELD(num) = colchoice .and. asc(colchoice) > 0
           IF TYPE('&colchoice') = 'N'
               x = .T.
               y = .T.
               AVERAGE &colchoice ALL TO sumtotal
               ? '     The average of that column is'
               ?? sumtotal
               ?
               ?
               WAIT
           ENDIF
        ENDIF
        num = num + 1
        IF asc(field(num)) = 0
           y = .T.
           num = 1
```

```
            colchoice = '              '
        ENDIF
        ENDDO
    ENDDO
    choice = ' '
ENDIF
IF choice='C'
    *
    * To do Operational Analysis of a table's column
    * This means we will calculate the mean, range,
    * standard deviation, and the standard error of the mean
    *
    DO TABLOPT
    tablechoice = UPPER(tablechoice)
    tablechoice = TRIM(tablechoice)
    IF tablechoice = 'Q' .OR. ASC(tablechoice) = 0
        RETURN
    ENDIF
    USE &tablechoice
    nonumeric = .T.
    i = 1
    DO WHILE ASC(FIELD(i)) # 0
        column = FIELD(i)
        IF TYPE('&column') = 'N'
            nonumeric = .F.
        ENDIF
        i = i + 1
    ENDDO
    IF nonumeric
        ?
        ? ' The table you have chosen has no numeric columns'
        ?? ' to average.'
        ?
        ?
        WAIT
        RETURN
    ENDIF
    CLEAR
    DISPLAY OFF ALL
    ?
    ?
    ?
    colchoice = '              '
    @ 22, 10 SAY ' Choose the column you wish analyzed:'
    x = .F.
    * if column "numeric" do operation
    num = 1
    DO WHILE .NOT. x
        @ 22,46 GET colchoice
        READ
        colchoice = TRIM(UPPER(colchoice))
        y = .F.
        DO WHILE .NOT. y
        IF FIELD(num) = colchoice .and. asc(colchoice) > 0
          IF TYPE('&colchoice') = 'N'
              x = .T.
              y = .T.
              SUM &colchoice ALL TO sumtotal
              counter = RECCOUNT()
              mean = sumtotal / counter
              sumsquares = 0
              sumvar = 0
              GOTO TOP
              number = '1'
              j = 1
              DO WHILE j <= counter
                  M&number = &colchoice
                  IF j = 1
                      biggest = &colchoice
                      smallest = &colchoice
```

```
            ENDIF
            IF M&number > biggest
               biggest = M&number
            ENDIF
            IF M&number < smallest
               smallest = M&number
            ENDIF
            sumsquares = sumsquares + (M&number ** 2)
            sumvar = sumvar + ((M&number - mean) ** 2)
            j = j + 1
            SKIP 1
            number = STR( VAL(number) + 1)
            number = LTRIM(number)
         ENDDO
         SORT TO temp2a.dbf ON &colchoice
         range = biggest - smallest
         z = sumsquares - ((sumtotal ** 2) / counter)
         standev = SQRT( z / (counter - 1))
         standerror = standev / (SQRT(counter))
         var = sumvar / (counter - 1)
         standev2 = SQRT(var)
         ?
         ?
         ? 'The column of your choice was :          '
         ?? colchoice
         ?
         ? 'The mean of the column was:          '
         ??mean
         ?
         ? 'The range of the column was:         '
         ??range
         ?
         ? 'The standard deviation was:          '
         ??standev
         ?
         ? 'The standard error of the mean was: '
         ??standerror
         ?
         ? 'The variance of the column was:      '
         ??var
         USE temp2a
         n = counter / 4
         GOTO TOP
         SKIP (n - 1)
         q = &colchoice
         SKIP n
         r = &colchoice
         SKIP n
         s = &colchoice
         SKIP n
         t = &colchoice
         ?
         ? '                    Quartiles          ',colchoice
         ? '                   -----------       -----------'
         ? '                       25%       ',q
         ? '                    50%(mode)    ',r
         ? '                       75%       ',s
         ? '                      100%       ',t
         ?
         ?
         erase temp2a.dbf
         WAIT
      ENDIF
   ENDIF
   USE &tablechoice
   num = num + 1
   IF .NOT. y
      IF asc(field(num)) = 0
         y = .T.
         num = 1
```

```
                colchoice = '          '
              ENDIF
            ENDIF
            ENDDO
        ENDDO
    ENDIF
    IF choice='D'
        *
        * To count the number of records in a table
        *
        DO TABLOPT
        tablechoice = UPPER(tablechoice)
        tablechoice = TRIM(tablechoice)
        IF tablechoice = 'Q' .OR. ASC(tablechoice) = 0
            RETURN
        ENDIF
        USE &tablechoice
        numrecords = 0
        CLEAR
        COUNT ALL TO numrecords
        @ 12,15 SAY ' The number of records in '
        ?? TRIM(tablechoice)
        ?? ' is '
        ?? numrecords
        @ 21,10 SAY ' '
        WAIT
        choice = ' '
    ENDIF
  a = .t.
ENDDO
*************************************************************************
* Program: TABLOPT.prg
* Purpose: Program presents choice menu to the user for
*          him to choose table catagory, then gives him
*          the option of what table he wants and makes
*          that table a public variable.
*************************************************************************
*
PROCEDURE TABLOPT
CLEAR
PUBLIC tablechoice, thrdchoice
SET HEADING OFF
*
* Draw a menu box
*
@ 1,2 to 22,78
@ 5, 23 SAY '  TABLE  CATEGORIES  AVAILABLE'
@ 6, 23 SAY '--------------------------------'
@ 8, 15 SAY 'Please choose from the list below:'
@ 10,19 SAY '(F)    FIRE SUPPORT tables'
@ 11,19 SAY '(I)    INTANGIBLES tables '
@ 12,19 SAY '(L)    LOGISTICAL tables '
@ 13,19 SAY '(T)    TACTICAL MANUEVERS tables '
@ 14,19 SAY '(U)    UNIT HISTORY tables '
@ 15,19 SAY '(Y)    YOUR OWN CATEGORY tables '
@ 16,19 SAY '(Q)    to QUIT or LEAVE this menu'
@ 20,20 SAY 'Enter your selection'
  thrdchoice = ' '
  tablechoice = ' '
DO WHILE .NOT. UPPER(thrdchoice)$'FILTUYQ'
  @ 20, 43 GET thrdchoice
  READ
ENDDO
thrdchoice = UPPER(thrdchoice)
IF thrdchoice ='Q'
  CLEAR
  tablechoice = ' '
  thrdchoice = ' '
  RETURN
ENDIF
```

```
* display possible tables to user to allow him to pick one
DO CASE
    CASE thrdchoice = 'F'
      CLEAR
      USE FSUPPRT
      @ 5, 25 SAY ' FIRE SUPPORT TABLES PRESENT'
      r = 8
      c = 0
      DO WHILE .NOT. EOF()
           IF c > 60
               r = r + 1
               c = 0
           ENDIF
           entry = FSTABLES
           @ r,c SAY entry
           SKIP
           c = c + 15
      ENDDO
      @ 19,15 SAY 'To leave this selection screen, type "Q"'
      @ 21,15 SAY 'Please enter the table of your choice:'
      tablechoice ='                '
      STORE .F. TO Y
      DO WHILE  .NOT. Y
        @ 21, 53 GET tablechoice
        READ
        STORE UPPER(tablechoice) to z
        z = TRIM(z)
        IF z = 'Q' .or. ASC(z) = 0
           DO TABLOPT
           RETURN
           tablechoice = ' '
           Y = .T.
        ENDIF
        LOCATE FOR FSTABLES = z
        Y = FOUND()
      ENDDO
    CASE thrdchoice = 'I'
      CLEAR
      USE INTANGIB
      @ 5, 25 SAY ' INTANGIBLE ITEMS TABLES PRESENT'
      r = 8
      c = 0
      DO WHILE .NOT. EOF()
           IF c > 60
               r = r + 1
               c = 0
           ENDIF
           entry = INTABLES
           @ r,c SAY entry
           SKIP
           c = c + 15
      ENDDO
      @ 19, 15 SAY 'To leave this selection screen, type Q'
      @ 21, 15 SAY 'Please enter the table of your choice:'
      tablechoice ='                '
      STORE .F. TO Y
      DO WHILE  .NOT. Y
        @ 21, 53 GET tablechoice
        READ
        STORE UPPER(tablechoice) to z
        z = TRIM(z)
        IF z = 'Q' .OR. ASC(z) = 0
           thrdchoice = ' '
           DO TABLOPT
           RETURN
           Y = .T.
        ENDIF
        LOCATE FOR INTABLES = z
        Y = FOUND()
      ENDDO
```

```
CASE thrdchoice = 'L'
   CLEAR
   USE LOGISTIC
   @ 5, 25 SAY ' LOGISTICAL TABLES PRESENT'
   r = 8
   c = 0
   DO WHILE .NOT. EOF()
        IF c > 60
            r = r + 1
            c = 0
        ENDIF
        entry = LGTABLES
        @ r,c SAY entry
        SKIP
        c = c + 15
   ENDDO
   @ 19,15 SAY 'To leave this selection screen, type "Q"'
   @ 21, 15 SAY 'Please enter the table of your choice:'
   tablechoice ='                '
   STORE .F. TO Y
   DO WHILE  .NOT. Y
      @ 21, 53 GET tablechoice
      READ
      STORE UPPER(tablechoice) to z
      z = TRIM(z)
      IF z = 'Q' .OR. ASC(z) = 0
         DO TABLOPT
         RETURN
         thrdchoice = ' '
      ENDIF
      LOCATE FOR LGTABLES = z
      Y = FOUND()
   ENDDO
CASE thrdchoice = 'T'
   CLEAR
   USE TACTICS
   @ 5, 25 SAY ' TACTICAL TABLES PRESENT '
   r = 8
   c = 0
   DO WHILE .NOT. EOF()
        IF c > 60
            r = r + 1
            c = 0
        ENDIF
        entry = TATABLES
        @ r,c SAY entry
        SKIP
        c = c + 15
   ENDDO
   @ 19,15 SAY 'To leave this selection screen, type "Q"'
   @ 21, 15 SAY 'Please enter the table of your choice:'
   tablechoice ='                '
   STORE .F. TO Y
   DO WHILE  .NOT. Y
      @ 21, 53 GET tablechoice
      READ
      STORE UPPER(tablechoice) to z
      z = TRIM(z)
      IF z = 'Q' .OR. ASC(z) = 0
         DO TABLOPT
         RETURN
         thrdchoice = ' '
      ENDIF
      LOCATE FOR TATABLES = z
      Y = FOUND()
   ENDDO
CASE thrdchoice = 'U'
   CLEAR
   USE UNITHIST
   @ 5, 25 SAY ' UNIT HISTORY TABLES PRESENT'
```

```
      r = 8
      c = 0
      DO WHILE .NOT. EOF()
           IF c > 60
               r = r + 1
               c = 0
           ENDIF
           entry = UHTABLES
           @ r,c SAY entry
           SKIP
           c = c + 15
      ENDDO
      @ 19,15 SAY 'To leave this selection screen, type "Q"'
      @ 21, 15 SAY 'Please enter the table of your choice:'
      tablechoice ='                '
      STORE .F. TO Y
      DO WHILE  .NOT. Y
         @ 21, 53 GET tablechoice
         READ
         STORE upper(tablechoice) to z
         z = TRIM(z)
         IF z = 'Q' .OR. ASC(z) = 0
            DO TABLOPT
            RETURN
            thrdchoice = ' '
         ENDIF
         LOCATE FOR UHTABLES = z
         Y = FOUND()
      ENDDO
   CASE thrdchoice = 'Y'
      CLEAR
      USE YOUROWN
      @ 5, 25 SAY ' USER DEFINED TABLES PRESENT'
      r = 8
      c = 0
      DO WHILE .NOT. EOF()
           IF c > 60
               r = r + 1
               c = 0
           ENDIF
           entry = YRTABLES
           @ r,c SAY entry
           SKIP
           c = c + 15
      ENDDO
      @ 19,15 SAY 'To leave this selection screen, type "Q"'
      @ 21, 15 SAY 'Please enter the table of your choice:'
      tablechoice ='                '
      STORE .F. TO Y
      DO WHILE  .NOT. Y
         @ 21, 53 GET tablechoice
         READ
         STORE UPPER(tablechoice) to z
         z = TRIM(z)
         IF z = 'Q' .OR. ASC(z) = 0
            DO TABLOPT
            RETURN
            thrdchoice = ' '
         ENDIF
         LOCATE FOR YRTABLES = z
         Y = FOUND()
      ENDDO
ENDCASE
*
SET HEADING ON
RETURN
```

APPENDIX G

USER TUTORIAL WITHIN PROGRAM

USE the CURSOR KEYS, PGUP, or PGDN to MOVE thru the tutorial.
USE the ESC KEY to EXIT the tutorial.

WELCOME to the database program developed to store data from the National Training Center. This program is designed to provide a novice user with the ability to manipulate data, that has been previously collected during a rotation, with ease and efficiency.

The idea of the program is for the user to operate through a series of menus to accomplish a specific task. At any time, should the user feel uncomfortable with his selection or wish to change it, he may enter a 'Q' in the choice selection box and return to the previous menu. This process will ultimately place the user at the opening menu where he may continue on a different path or exit the system.

Before we begin, let us define some common terms that will facilitate the user's understanding of the descriptions used throughout the program. A Table refers to the format which we use to logically store information, and is synonomous with the common usage of the term "file". Our tables are subdivided into different columns of information or "fields". A single line entry is called a "record" and it has one and only one value for each column or field within the table it is assigned to.

The program allows the user to perform the following actions via menu selection choice options:

CREATE - This allows the user to make a new table complete with column headings and typing to store new data of his choice.
ADD - This option provides the user with the ability to add a record of information to an existing table.
CHANGE - This option allows the user to change or update information previously stored in a table.
REMOVE - The allows for the removal of a specific record of information within a table.
DISPLAY - This option allows the user to print the contents of a table to the screen.
LINK - This option allows the user to combine the data from two tables of his choice into a new table. This action is done by combining the tables upon the columns which are similar to both, then printing all the different record entries using the data from each of the old tables. The user is cautioned to use this option judisously as this combination process can create a very large table.
SELECT - This option can be used by the user to create a new table which has only a portion of the number of columns that some original table had.
FIND - This option allows the user to locate a particular record within a known table and return that record to the screen. It can also be used to show membership within a table of a value corresponding to one of the fields.
MATHEMATICAL OPERATIONS - Using numerical entries within a table, the user can use this option to sum or average the data in that column. The analysis option allows the user to perform some basic operational analysis functions upon the values of a column. He may also use this choice to count the number of records in a table.
TUTORIAL - The option that you have currently selected, is recommended as a learning tool to acquaint the user with the various capabilities of this program.

FILE MAINTENANCE OPERATIONS - These choices allow the user to erase a table from memory (thereby destroying the contents forever), restore data from a disk to the tables in use (in a situation of catastophic failure), or save a the data tables to memory.

The OPENING MENU gives the user five choices. He can use the TUTORIAL to receive instructions concerning the program, as you have done, he can MAKE or CHANGE data within a table, he can MANIPULATE or LOCATE data elements, he can accomplish TABLE MAINTENANCE operations to restore or save the contents of the data tables, or he can QUIT and leave the program. The QUIT command places him back at the DOS prompt outside the DBASE III system.

Lets try the second option since you've already tried the first. The MAKE or CHANGE data option leads the user into another menu of five choices. Here the user is asked whether he wants to CREATE a table to display new information, ADD or CHANGE a record in an existing table, REMOVE a record from an existing table, DISPLAY a table, or QUIT this menu and return to the opening menu. To see a further explanation of the steps corresponding to these choices, please use the PGDN key until you reach the appropriate title heading (i.e. CREATE).

The third option of the opening menu gives the user another menu that is similar to the menu described in option two. Here the user has the opportunity to pick from five other choices. Using this menu he may LINK two existing tables together to get a third which is the combination of those two originals, he may SELECT a specific field or fields from an existing table and reproduce those into a new table, he may FIND a specific piece of data in a table, he may do MATHEMATICAL OPERATIONS on a table or the numeric columns in that table, or he may QUIT this menu and return to the opening menu again. As before, should you wish to follow a particular option for further details, please refer to the portion corresponding to the option heading.

The fourth option from the opening menu allows the user to perform FILE MAINTENANCE OPERATIONS on the database tables. Using this option, the user has the capability to ERASE a table from the system memory, BACKUP all his tables from a disk using one of the internal disk drives on his AT computer, SAVE the database tables to a memory disk, or QUIT and return to the opening menu. Again, to see further details on any of these options, please refer to them directly.

The final option of the opening menu is the QUIT command. Using this command the user returns himself to the DOS prompt outside this SYSTEM. In the process of exiting the system, the user is given the opportunity to perform file maintenance operations should he/she had changed anything. The user is required to make a conscious choice to continue in order to enforce good file management.

CREATE

This option allows the user to design a table of his own. Using this choice following the choice of the second option from the opening menu, the program will interactively ask the user what the name of the new table he is creating is, wait for the answer, then ask him to identify the names of the columns within the table (these are also known as fields). Should he request to use a name that already is being used, the program will ask him if he wants to overwrite the old table with this new data. If he enters 'Y' the program will continue on. A 'N' will place him at a spot where he has the option to relocate his table or go back at the start where he may use another name for his table. The program will ask after each column name, what TYPE that field is. The type can be either a character (which is like letters or names), an numeric (a number), a date (ex. 04/12/78), a logical (a true or false value), or a memo (used to record a long series of sentences). After the user decides the type, he

will be asked to decide how long he wants the data entry in the column to be: characters can be up to 254 characters long, numerics up to 15 digits, dates are stored as the example indicates MM/DD/YY, logicals are either .T. or .F., and memos can range from 512 to 4096 characters in length. Should the user pick numeric as his choice, he will be asked to decide how many places after the decimal point should be displayed. This can be any number between 0 and 13. Press the ENTER key after each entry to move through each highlighted area.

To stop entering any more fields the user simply presses the ENTER key when queried for a blank column name. The program will then ask the user if he is sure that this is the way he wants his new table. Following the menu directions he can change any of the entries, or he can press the ENTER key to save this table's form.

The next question that the user is asked is whether or not he wants to enter data directly into his new table or wait. Entering a 'Y' here allows the user to enter info directly in. The screen will then present a highlighted area for each part of the record so the user may enter the necessary data. Blanks may be maintained in areas by simply using the ENTER key when you reach that point. Again, to exit this stage simply press the ENTER at the beginning of a record or a better method is to press the CTRL and END keys simultaneously at the end of the last record entry.

The next menu allows the user to catagorize his new table under an appropriate heading. The choices here reflect the five established catagory headings plus the heading YOUR OWN tables which are user defined. The user is forced to catagorize his table at this time, so should he find that his table is in error, he may use the CHANGE option to modify individual records or the ERASE option to destroy the file. This completes the CREATE option and the user is returned to the appropriate menu from which he started.

ADD

The ADD option allows the user to add a record of information to the table of his choice. Upon entering this choice from the conformation menu the user is presented with the category table choices from which he chooses the category of his table. Having chosen the category, he is then presented with the tables in that category from which he can make his table choice. In the event that he has made an incorrect choice or wishes to change, he may enter a 'q' to leave the category and choose another. He may continue in this vain to exit this option choice.

Assuming that he has picked a table, the user is now presented with the different columns of the record and a highlighted area to the right of each column where he can enter the data. The length of the highlighted area shows how long the entry can be and the user will be moved to each sequentially by pressing the enter key or filling the preceding highlighted area completely. Notice at the top of the page there is a number indicating the corresponding record number that the user is working on.

The user can follow the instructions provided in the menu block to complete his actions or he may press the ENTER key at the beginning of the first highlighted block to exit this addition process. Upon exiting, the user will be placed back at the option menu from which he started.

CHANGE

The CHANGE operation allows the user to modify a record which is already in existance. The user has reached this choice by following the same procedure used to reach ADD. At the point where the menu asked for a choice between ADD and CHANGE, the user chose CHANGE. The user is then confronted with the category choice menu. After picking the appropriate category that his table resides in, he then enters the table's name in the space provided. At this point should he wish to return to the category

choices, he can enter a "Q" in the space reserved for the table name.

Assuming that the user wishes to continue, after picking the table, the records of that table are displayed upon the screen. Using the directional arrow keys, he can move through the records and find the one he wishes to change. Then using the directional keys he can move to the appropriate value and overwrite it in the space provided. Typing the ENTER key stores the value in place. Should the user traverse the entire length of the table, he will be given the option to add data to the table. The user can accomplish this by entering a "Y" after the question. Then as in the add operation he can input the data in the space provided. He exits this mode by typing the ENTER key on a blank field. This will place him at the last value entered where he may perform any changes he wishes or follow the menu information and leave the option (CTRL - END). Should he wish to not save these changes he can abort by pressing the ESC key. In either case, the user will return to the original menu where he can pick another MAKE or CHANGE option.

DELETE

The REMOVE option is reached by entering choice "C" within the MAKE or CHANGE menu. This choice allows the user to delete a record from an existing table. Once the user picks this option, he is confronted with another menu which confirms that he wishes to delete a record from a table. Picking the letter "D" continues the action and places the user at the category choice menu. After picking the appropriate category and table name the user is able to begin the operation.

At this point the user is presented with a display of the table of his choice with each record numbered. At the base of the screen, the user is provided a area to enter the respective number of the record that he wishes to delete. Error checking ensures the the input number is within the proper boundaries. Should he wish to not delete any record he is given the last chance to exit by entering a "Q" in the space provided for the record number. Once a number is entered the user can use the SPACE bar to complete the area or press the ENTER key. In each case the record will be deleted and by pressing another key of his choice, he will return to the original menu.

DISPLAY

The DISPLAY option is the last choice of the MAKE or CHANGE menu. This option allows the user to list the contents of the table of his choice or display the structure of the table. Once this option has been chosen, the user is confronted with the menu to choose between displaying of the contents or the structure. In either case once the user chooses a "D" or an "S" ("Q" returns him to the original menu), he is placed at the category choice menu. After choosing the category, he is then required to enter the name of the table he desires to display.

If the user had picked a "D" the table contents will be listed on the monitor screen. Should the table be longer than twenty lines, including wrap arounds from wide tables, the user must press a key to continue the display. Once the display is completed, the user is returned to the original menu.

If the user had picked an "S" the different field names of the table will be displayed on the screen. Next to each field name is its respective type, field width, and if it is a numeric, the number of places to the right of the decimal point. By pressing a key of his choice, the user can return to the original menu.

LINK

The LINK operation is often refered to as a join operation. This is because the operation joins the data of two tables together based upon a common field. It is important to note that the operation joins on a common field regardless of whether

there are common values within that field. In the best case each value of one table has a record in another with that same value, and the new table is the length of the original tables. In the worst case each value from the first table is paired with a value from the second. In this case, the length of the new table is the length of the first TIMES the length of the second. It is for this reason that the user is cautioned when using this operation.

Once the user has decided to use this function, he is asked to pick the first table to be used in the operation. He does this by typing the appropriate category choice and table name. The program will then echo the name of the table which he has picked. He's then given the opportunity to choose the second table in a similar fashion. In either case entering a Q in the category choice places him back at the original menu. Once he has chosen both tables the program will ask him to specify the name of his new table. A similar error checking scheme is performed as in the CREATE function and if the table name is not in use, the operation will proceed. Should the name be used the program will place the user back at the start point.

If the table name is not in use, the user is then queried as to the category which he wishes to store his new table under. Once this choice has been made, the operation will continue and the actual linking of the data elements conducted. At the completion of this operation the user will be asked to press any key to place him back at the original menu.

SELECT

The SELECT option allows the user to create a table using an existing table. The user does this by specifying the fields of data which he wishes to copy from the old table into his new one. Once he has chosen to continue the operation, he is asked to choose the appropriate table category and name. After picking the table name, the user is provided with a listing of the table to help him determine which fields he wishes to use. At the bottom of the screen, the program will ask him which fields' by name, he wishes to copy. Only a "Y" response copies the field, any other response is interpreted as a "N".

After he has chosen the fields he wishes to copy, the program will ask the user to specify the table name that the data will be stored under. Here again a checking operation similar to the LINK operation is conducted. If the table name has been used, the user is returned to the start point to try again. If the name is valid, then the user will be asked which category to store the table under. When the operation is complete, the user will be asked to press any key to return to the original menu.

FIND

The FIND operation is used in two situations. First, the Find operation can be used to locate records within a table that contain a specific value. The other reason is to show membership of a value within a table. To utilize the FIND operation, the user must know what value he is looking for and which table is to be scanned. After confirming that he wishes to proceed with this option, the user is asked whether he wishes to retain a table with the information returned from the FIND operation. This allows the user to maintain a permanent record suitable for a query operation. If the user answers "Y" then he must specify a table name under which the data will be stored. The table is automatically conformed to store the data (provided the name passes the error checking scheme). Then the user chooses the new table storage category.

After he has completed the above or should he have answered the question "N", he must pick the appropriate table category and table name. He is then asked to specify the data value which he desires to find an occurrence of in the table. The program will check each field for the value and return those records which contain the value. Should no record be found, a message confirming this will appear on the screen. At the completion of

the operation, the user will be asked to press any key to return to the original menu. If the storage option was chosen, the records displayed will be stored in the new table.

MATHEMATICAL OPERATIONS

This option provides the user the capability to accomplish some basic mathematical operations on a column of data. This option is reached through entering a "D" in the MANIPULATE and LOCATE menu. Upon choosing this option, the user is presented with another menu which points out his operation choices. The user may SUM the elements of a column, AVERAGE the elements of a column, perform an ANALYSIS of the elements of the column, or he may COUNT the number of records within a table. In each of the first three options the user is allowed to only attempt the operations on a data column of type numeric. The final operation can be used upon any table.

The same procedure applies to each of the first three choices, so a general description will be given. In each case, once the user has made a selection of the operation, he is queried for the table category and name. Once the user has identified the table of his choice, a listing of the table is displayed for user convenience. He is then asked to denote the name of the column upon which he wants to perform the operation. Should he pick a column which is not typed numeric, no action will take place and the entry will be ignored.

When the user has selected a proper data column the operation will act. The SUM operation will return a sum total of all entries in the column. The AVERAGE operation will return a value which represents an average for the data column. The ANALYSIS will return values for the mean, range, standard deviation, standard error of the mean, and a quartile listing. The quartile listing shows the data value which corresponds to a percentile evaluation of the sorted data values. Once these values have been displayed on the screen, the user will be asked to press any key to return to operation choice menu.

The option to calculate the number of records within a table or COUNT begins in a similar manner. The user is queried for the table category and name. Once the table name has been identified, the program returns a value that represents the number of records present in the table. The user is then asked to press any key to return to the operation choice menu. At this menu the user can continue working or enter a "Q" to return back to a previous higher level menu.

FILE MAINTENANCE

This option is called in two places. The user can reach this menu by choosing the fourth option from the opening menu, or he can answer "Y" during the QUIT procedure to exit the program. In either situation, the user is presented with a menu of choices which allow him to accomplish the following: ERASE a table from memory, RESTORE data from a floppy disk to his tables in main memory, or SAVE a copy of the changes to his tables on a floppy disk and create a backup for the system.

The ERASE operation erases a table from memory and also removes the table name from the appropriate category. The user is asked to select a table category and name as before. Once the user has selected a table name, the program will ask him again if he wishes to complete this action. Any answer other than a "Y" returns the user to the operation choice menu. A "Y" erases the table and the data is lost from memory. The user is then asked to press any key to return to the operation choice menu.

The RESTORE operation allows the user to recopy information from a backup floppy into main memory. As with any RESTORE operation the information is only as current as the last backup save procedure. Once the user chooses this option he is asked again if he wishes to complete this. The significance of this question is that the RESTORE operation will overwrite any new data within the tables and lose any additions done after the backup save was

done. Here again, any answer other than a "Y" returns the user to the operation choice menu.

Should the user wish to continue, he is asked to place his backup floppy into the "B" drive of his system. When the floppy has been placed in the disk drive, the user presses any key to continue. Once the operation is complete, the user presses any key to return to the operation choice menu.

The SAVE operation is just the reversal of the RESTORE operation. In this situation the user is moving a copy of his working tables onto a floppy backup. The user can use any formatted disk to be the backup medium and the "B" drive is used in this procedure also. Once this operation has been completed, the user is asked to press any key to return to the operation choice menu.

****

This completes the tutorial. If you have questions that have not been answered within the context of this file, they probably involve the microcomputer that you are using. For more information about the program concept, please see the user's manual within the thesis document. We wish you much success in your endeavors.

PRESS THE <ESC> KEY AT THIS TIME TO EXIT THE TUTORIAL OR USE THE MOVEMENT KEYS TO MOVE BACK UP TO REREAD ANY SECTION.

APPENDIX H

OTHER IMBEDDED PROGRAMS USED BY THE DBM PROGRAM

```
rem  Program: Erasor1.bat
rem  Purpose: Eliminates a temporary file created during copy process
ERASE C:templa.DBF

rem  Program : Savprog.BAT
rem  Purpose : This program provides the user with the capability
rem            to store a copy of his files to a backup floppy disk
copy c:fs*.dbf B:
copy c:in*.dbf B:
copy c:logistic.dbf B:
copy c:lg*.dbf B:
copy c:ta*.dbf B:
copy c:unithist.dbf B:
copy c:uh*.dbf B:
copy c:yourown.dbf B:
copy c:yr*.dbf B:

rem Program : SAVINTO.BAT
rem Purpose : This program allows the user to restore his working files
rem           from a backup source.
copy b:fs*.dbf c:
copy b:in*.dbf c:
copy b:logistic.dbf c:
copy b:lg*.dbf c:
copy b:ta*.dbf c:
copy b:unithist.dbf c:
copy b:uh*.dbf c:
copy b:yourown.dbf c:
copy b:yr*.dbf c:
```

APPENDIX I
ADDITIONAL EVALUATION CRITERIA

These evaluation sheets present one possible means to collect the required data to fill some of the tables within the new database. These are intended to allow the evaluator to make simple, objective evaluations of a specific area of interest. The sheets are grouped by data subject area and are shown in a tentative formatted display. Each display will represent a different evaluation sheet.

1. FIRE SUPPORT

This sheet will require a summation of the missions fired for a given "battle."

MissionNum	Total Missions fired	Total 4.2" Missions	Total 155MM Missions	Total 8" Missions	Total Number of Fratricides

2. INTANGIBLES

These sheets are intended to be used by the individual unit evaluator to record aspects of certain areas of combat operations during the preparation phase of each "battle."

Mission Number	Unit	Boresight (Y/N)	Distance	Tactical Feeding (Y/N)	Sleep Plan (Y/N)	Percent Security	Tactical Refuel (Y/N)

Mission Number	Bde Order Time	Bn Order Time	Co Order Time	PLT Order Time	MOPP Level	OPORD Paragraph Evaluation

Mission Number	Temp	Percent Humidity	Percent Moonlight (night)	Wind Speed	Wind Direction	Visibilty Distance

3. LOGISTICS

The first sheet will require units to ensure proper completion of the fuel/refuel allocation sheet presently in use. The next sheets will used by the respective administrative area evaluator.

Unit Identification	Total fuel for Trucks	Total fuel for Tanks	Total fuel for APCs	Total fuel for Bradleys	Total fuel for TOWs

	Miles per Gal trucks	Miles per Gal Tanks	Miles per Gal APCs	Miles per Gal Bradleys	Miles per Gal TOWs

Mission Number	Unit	Supply Route Distance	Time for Trip	Emergency Request

Mission Number	Total Number Casualties	Number Evacuated Bn Aid Stat.	Number Evacuated Field Hospital

4. TACTICAL MANEUVERS

These sheets are intended to be used by the individual unit evaluator to record aspects of certain areas of combat operations during the maneuver phase of each "battle."

Mission Number	Unit	Agent Used	Agent Reported	Unit Prewarned (Y/N)	Masking Time	Number Casualties	Area Prediction Made

Mission Number	Unit	Line of Departure Time	Time Late LOD	Distance From AA To LOD

5. UNIT HISTORY

This questionaire is to be filled out prior to the unit's arrival at the National Training Center. The questionaire can be handed out at the

pre-rotation brief at the uniyt's home station or sent to the unit at a more convenient time.

Unit ID Number	Type of Unit	Home Station Location	Unit Strength	Percent Officers Assigned	Percent Enlisted Assigned	Date Notified	Rotation Date

Number of Training Cycles Before Rotation Date

Number 4 Man Tk Crews	Certified 4 Man Tank Crews	Number 3 Man Tk Crews	Certified 3 Man Tank Crews	Number Full Infantry Squads	Number Other Squads

Unit Position	Time In Psn	Number Rotation in Psn	Military Schools Attended	Previous Rotation Experience

APPENDIX J

CONTRACTUAL AGREEMENT WITH SPONSOR

SECTION 1 PURPOSE

1.1 Request. This outline will serve as an agreement in principle as to the functions of the thesis project. This also serves as an information tool to show the depth of detail that the project undertakes.

1.2 Action. The sponsor is requested to make any modifications that are deemed appropriate and return such to the sender. Should no changes be necessary, the outline will serve as an information paper concerning the project that will be delivered to the sponsor.

SECTION 2 SYSTEM SUMMARY

2.1 Background. The National Training Center represents the most complete training facility ever devised for use within the United States Army. Since 1981 when the facility was opened to rotational units, vast amounts of information relating to nearly every phase of a unit's performance have been generated as the unit performs under the watch of the evaluation group. A typical rotation lasts approximately four weeks with week one encompassing the drawing of tactical vehicles and pre-mission tasks. Week two and three encompass the actual force-on-force (10 days) and live fire exercises (5 days). During the final week the unit returns its borrowed vehicles and key personnel receive their final outbriefs. The units return to their home stations with a take home evaluation package. This package highlights the tactical play of each mission in terms of the seven operating systems. Several data tables are presented at the completion of each mission description which report the unit's losses in terms of vehicles and personnel and their corresponding kill ratios against the opposing forces.

2.2 Objectives. The overriding objective of the thesis is to provide a system which can relate information from several significant areas of interest to the unit's kill ratio for a given mission. The design objectives of this thesis are to furnish the following to the sponsoring office:

a. to provide a database framework to allow the storage of information generated by the rotational unit so as to give a more complete picture of the unit's performance.

b. to provide an information management system to allow the user to manipulate the data in a meaningful manner.

c. to provide a system to accomplish the above in a timely and efficient manner.

d. to provide to the sponsor (the intended user) the location of known collected data and in areas where the data is not collected, a recommended method of data collection.

2.3 Existing System. Currently the tactical information is collected as per contract with SAIC. SAIC has also won a contract to update the existing system. At present the historical data from rotations past has been stored on magnetic tapes. These tapes are under operational control of CATA, Ft. Leavenworth. The tapes are stored at ARI Monterey. The vehicle logistical data is being collected by the private contractor DYNALECTRON CORP. All other information is manually collected but not maintained after the rotation. Several areas of interest (i.e. unit history, etc.) are not collected.

2.3.1 Existing Software/Hardware. At present the sponsoring office has limited desktop microcomputer facilities. These assets are IBM compatible.

2.4 Present Procedures. To utilize any of the tactical data collected to date, an agency must first obtain authorization from CATA. Once permission has been granted, the request is then forwarded to ARI Monterey for completion of the request. Upon completion of the search by ARI, a "scrubbed" version of the infromation will be sent to the requestor. Historical data concerning the vehicles is incomplete and misleading due to faulty procedures by the previous contractor. Other information can be obtained through specific unit contact to access personnel data and after action reports. Units are under no pressure to cooperate with such requests and grant these requests on a case by case basis. The bottomline is that information access is difficult at best.

2.5 Proposed System. The proposal is made utilizing existing conditions or conditions which are presently under contract to exist in the near future. It also understood that the sponsoring office is in the process of procuring an IBM PC AT.

2.5.1 Information Areas. The new contract with SAIC stipulates the establishing of a work station at each home station of a rotational unit. These workstations allow a unit to fully utilize the recorded tapes that will become part of the take home package. Tactical information as prescribed by the data locations within those tapes can be accessed from the apprpriate unit's tapes. Unit historical data and other applicable information can be obtained from the apprpriate unit. The improvement is that the request can be made in person since the unit is assigned to the same installation. Logistical data concerning the tactical vehicles is treated in a similar fashion as a completed DA Form 2406 and accessing that information requires a formal procedure. However, in this situationthe data is more reliable.

2.5.2 Information Entry. Since the system is designed to be a stand alone system, the information will have to be manually entered. The management system will attempt to make this procedure as easy as possible.

2.5.3 Access / Use. Because of the sensitivity of the information provided it is understood that the information will be used with discretion when unit identification becomes involved. However, by enlisting the unit's cooperation in gathering the information, the unit may "scrub" their respective data prior to release.

2.5.4 Output. The output will be limited to screen displays in an effort to help maintain discretionary use. Modification of the system to allow hardcopies can be nade by onsite personnel as the need develops.

2.5.5 Maintenance. The proposed system is envisioned to utilize a commercially available database management system. Specific documentation will also accompany any program design to facilitate maintenance operations.

LIST OF REFERENCES

1. Army Times. *Army Times Guide to Army Posts.* The Stackpole Company, Harrisburg, PA, 1966.

2. Simpson K. W. *The National Training Center: A Critique of Data Collection and Dissemination.* National War College, Washington, D.C., March 1985.

3. Government Accounting Office. *Army Training - National Training Center's Potential Has Not Been Realized.* GAO/NSAID-86-130, July 23, 1985.

4. Kroenke, David. *Database Processing, 2nd edition.* Science Research Associates, Inc., Chicago 1983.

5. Modell, M.E. *IEEE 1985 International Conference on Entity-Relationship Approach: The Entity-Relationship Approach as a tool for Application Analysis.* IEEE Computer Society Press, 1985.

6. Howe, D.R. *Data Analysis for Data Base Design.* Edward Arnold Publishers Ltd., Baltimore, MD, 1983.

7. Ullman, Jeffrey D. *Principles of Database Systems, 2nd edition.* Computer Science Press, Rockville, MD, 1982.

8. Chung, I., Nakamura, F., Chen, P.P. *A Decomposition of Relations Using the Entity-Relationship Approach, Entity-Relationship Approach to Information Modeling and Analysis.* ER Institute, 1981.

9. Ling, T.-W. *IEEE 1985 International Conference on Entity-Relationship Approach: A Normal Form for Entity-Relationship Diagrams.* IEEE Computer Society Press, Silver Springs, MD, 1985.

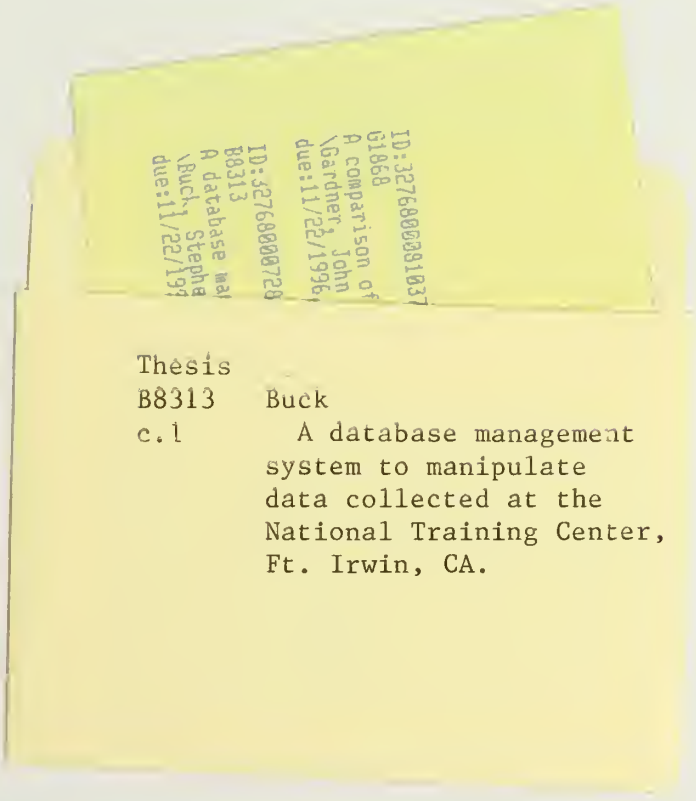
10. SAIC. *Requirements Design Specifications for a Prototype NTC Research Data Base System. Final Report.* Science Applications International Corporation, La Jolla, CA, July 1984.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Deputy Undersecretary of the Army for Operations Research Room 2E261, Pentagon Washington, D.C. 20310	2
4.	Commander US Army Training Support Center ATTN: ATIC - NC Fort Eustis, VA 23604-5166	2
5.	Directorate Of Combat Developments ORSA Branch ATTN: LTC G. S. Williams Headquarters, US Army Armor School Ft. Knox, KY 40121-5215	2
6.	Director U.S. Army TRADOC Operations Research Agency White Sands Missile Range, NM 88002	1
7.	Commander US Army TRADOC Analysis Center ATTN: Mr. Reed Davis Fort Leavenworth, KS 66027	2
8.	Bell Hall Library U.S. Army Combined Arms Center Fort Leavenworth, KS 66027	2
9.	Professor Samuel H. Parry, Code 55Py Department of Operations Reaserch Naval Postgraduate School Monterey, CA 93943	2
10.	Major John B. Isett, Code 52Is Department of Computer Science Naval Postgraduate School Monterey, CA 93943	1

11.	Curricular Office ATTN: Code 37 Department of Computer Science Naval Postgraduate School Monterey, CA 93943	1
12.	Director DCS - Training Headquarters, U.S. Army TRADOC Fort Monroe, VA 23651	1
13.	Commander USACAG ATTN: ATZL-TAL-N Fort Leavenworth, KS 66027-7000	1
14.	Cpt Stephen D. Buck 113 Andrew Drive Newtown, PA 18940	2

DUDLEY KNOX LIBRARY
GRADUATE SCHOOL
MONTEREY, CALIFORNIA



Thesis
B8313 Buck
c.1 A database management system to manipulate data collected at the National Training Center, Ft. Irwin, CA.

thesB8313
A database management system to manipula
3 2768 000 72867 9
DUDLEY KNOX LIBRARY